

Docket No.: 49657-921

#3  
LTyson  
03-22-01  
PATENT

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of

Toyohiko YOSHIDA

Serial No.:

Group Art Unit:

Filed: January 10, 2001

Examiner:

For: INSTRUCTION TRANSLATOR TRANSLATING NON-NATIVE INSTRUCTIONS  
FOR A PROCESSOR INTO NATIVE INSTRUCTIONS THEREFOR, INSTRUCTION  
MEMORY WITH SUCH TRANSLATOR, AND DATA PROCESSING APPARATUS  
USING THEM

**CLAIM OF PRIORITY AND  
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Commissioner for Patents  
Washington, DC 20231

Sir:

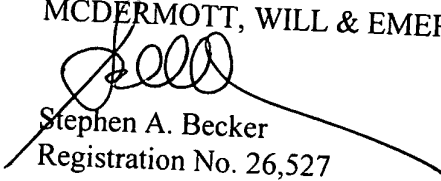
In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims the priority of:

Japanese Patent Application No. 2000-005062,  
filed January 13, 2000

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

  
Stephen A. Becker  
Registration No. 26,527

600 13<sup>th</sup> Street, N.W.  
Washington, DC 20005-3096  
(202) 756-8000 SAB:klm  
**Date: January 10, 2001**  
Facsimile: (202) 756-8087



日 本 国 特 許 庁

PATENT OFFICE  
JAPANESE GOVERNMENT

44657-921  
JANUARY 10, 2001  
YOSHIDA  
McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 1月13日

出 願 番 号

Application Number:

特願2000-005062

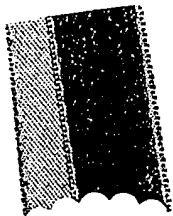
出 願 人

Applicant (s):

三菱電機株式会社



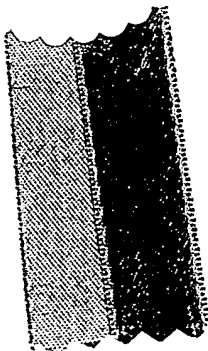
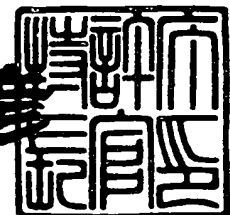
CERTIFIED COPY OF  
PRIORITY DOCUMENT



2000年 2月 4日

特許庁長官  
Commissioner,  
Patent Office

近藤 隆彦



出証番号 出証特2000-3004163

【書類名】 特許願

【整理番号】 520274JP01

【提出日】 平成12年 1月13日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/30  
G06F 9/455

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社  
社内

【氏名】 吉田 豊彦

【特許出願人】

【識別番号】 000006013

【氏名又は名称】 三菱電機株式会社

【代理人】

【識別番号】 100064746

【弁理士】

【氏名又は名称】 深見 久郎

【選任した代理人】

【識別番号】 100085132

【弁理士】

【氏名又は名称】 森田 俊雄

【選任した代理人】

【識別番号】 100091409

【弁理士】

【氏名又は名称】 伊藤 英彦

【選任した代理人】

【識別番号】 100096781

【弁理士】

【氏名又は名称】 堀井 豊

【選任した代理人】

【識別番号】 100096792

【弁理士】

【氏名又は名称】 森下 八郎

【手数料の表示】

【予納台帳番号】 008693

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 命令トランスレータ、トランスレータ付命令メモリおよびそれらを用いたデータ処理装置

【特許請求の範囲】

【請求項 1】 第 1 の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、前記第 1 の命令体系と異なる第 2 の命令体系の命令を記憶する命令メモリとともに用いられ、前記第 1 の命令体系の命令に変換して前記プロセッサに与えるための命令トランスレータであって、

前記プロセッサが実行すべき命令の、前記命令メモリにおけるアドレスを受けて前記命令メモリから対応する命令を読み出し、読み出された前記第 2 の命令体系の命令を前記第 1 の命令体系の命令に変換するための変換手段と、

前記変換手段により変換された後の前記第 1 の命令体系の命令を、前記命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、

前記プロセッサが実行すべき命令のアドレスを受けて前記保持手段を探索し、前記保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、前記変換手段の出力する命令と、前記保持手段に保持されていた、対応の前記第 1 の命令体系の命令とを選択的に前記プロセッサに出力するための選択手段とを含む、命令トランスレータ。

【請求項 2】 前記第 2 の命令体系は可変長命令体系であり、前記変換手段は、前記命令メモリから読み出された前記第 2 の命令体系の命令を、前記読み出された前記第 2 の命令体系の命令の命令長に依存した数の前記第 1 の命令体系の命令に変換するための可変長命令変換手段を含む、請求項 1 に記載の命令トランスレータ。

【請求項 3】 前記可変長命令変換手段は、前記命令メモリから読み出された前記第 2 の命令体系の命令を、前記読み出された前記第 2 の命令体系の命令の命令長に依存し、かつそれよりも長い長さの前記第 1 の命令体系の命令に変換するための第 1 の手段を含む、請求項 2 に記載の命令トランスレータ。

【請求項 4】 前記第 1 の命令体系の各命令は 1 または複数個のサブ命令を含み、前記第 1 の手段が変換する第 1 の命令体系の命令に含まれるサブ命令の個

数は、前記読み出された前記第2の命令体系の命令の長さに依存する、請求項3に記載の命令トランスレータ。

【請求項5】 前記変換手段は、前記命令メモリから読出された前記第2の命令体系の複数個の命令を、1つの前記第1の命令体系の命令に変換するための複数命令変換手段を含む、請求項1に記載の命令トランスレータ。

【請求項6】 前記第1の命令体系の各命令は、1または複数個のサブ命令を含むことが可能であり、

前記変換手段は、前記命令メモリから読出された前記第2の命令体系の複数個の命令を、前記複数個の命令の数に依存した数のサブ命令を含む、前記第1の命令体系の命令に変換するための手段を含む、請求項1に記載の命令トランスレータ。

【請求項7】 前記変換後の前記第1の命令体系の命令に含まれるサブ命令の個数が、前記複数個の命令の数と等しい、請求項6に記載の命令トランスレータ。

【請求項8】 第1の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、前記第1の命令体系と異なる第2の命令体系の命令を記憶する命令メモリとともに用いられ、前記第1の命令体系の命令に変換して前記プロセッサに与えるための命令トランスレータであって、

前記プロセッサが実行すべき命令の、前記命令メモリにおけるアドレスを受けて前記命令メモリから対応する命令を読出し、読出された前記第2の命令体系の命令を前記第1の命令体系の1または複数個の命令に変換する変換手段と、

前記変換手段により変換された後の前記第1の命令体系の命令を、前記命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、

前記プロセッサが実行すべき命令のアドレスを受けて前記保持手段を探索し、前記保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、前記変換手段の出力する命令と、前記保持手段に保持されていた、対応の前記第1の命令体系の命令とを選択的に前記プロセッサに出力するための選択手段と、

前記保持手段に保持されている命令を第1の条件および第2の条件のいずれか

で無効化可能なエントリとして保持するように前記保持手段を制御するための保持制御手段とを含む、命令トランスレータ。

【請求項 9】 前記第 1 の条件は前記保持手段による所定のアルゴリズムに基づくハードウェア制御による保持制御の条件であり、

前記第 2 の条件は前記保持手段からの外部からの明示的な無効化指示があったという条件である、請求項 8 に記載の命令トランスレータ。

【請求項 10】 前記保持制御手段は前記第 2 の条件で無効化が可能なエントリを無効化することなしに新たな命令を前記保持手段に保持できないときにアサートされる信号を出力する、請求項 8 に記載の命令トランスレータ。

【請求項 11】 第 1 の命令体系の命令をネイティブ命令として動作するプロセッサとともに用いられる命令メモリであって、

第 2 の命令体系の命令を記憶する命令記憶手段と、

前記命令記憶手段から出力される前記第 2 の命令体系の命令を、前記第 1 の命令体系の命令に変換して前記プロセッサに与えるための命令トランスレータとを含む、トランスレータ付命令メモリ。

【請求項 12】 前記命令トランスレータは、前記命令記憶手段から読み出すべき命令のアドレスに基づいて、前記第 2 の命令体系の命令を前記第 1 の命令体系の命令に変換する処理と、前記第 2 の命令体系の命令をそのまま出力する処理とのいずれかを選択的に実行するための手段を含む、請求項 11 に記載のトランスレータ付命令メモリ。

【請求項 13】 前記命令記憶手段からの読出時にアドレス変換を行なうためのアドレス変換手段をさらに含む、請求項 11 に記載のトランスレータ付命令メモリ。

【請求項 14】 前記命令トランスレータは、

前記プロセッサが実行すべき命令の、前記トランスレータ付命令メモリにおけるアドレスを受けて前記トランスレータ付命令メモリから対応する命令を読出し、読出された前記第 2 の命令体系の命令を前記第 1 の命令体系の命令に変換する変換手段と、

前記変換手段により変換された後の前記第 1 の命令体系の命令を、前記トラン

スレータ付命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、

前記プロセッサが実行すべき命令のアドレスを受けて前記保持手段を探索し、前記保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、前記変換手段の出力する命令と、前記保持手段に保持されていた、対応の前記第1の命令体系の命令とを選択的に前記プロセッサに出力するための選択手段とを含む、請求項11に記載のトランスレータ付命令メモリ。

【請求項15】 第1の命令体系の命令をネイティブ命令として動作するプロセッサと、

前記プロセッサが接続される信号転送手段と、

前記信号転送手段を介して前記プロセッサと相互接続されるトランスレータ付命令メモリとを含み、

前記トランスレータ付命令メモリは、

前記プロセッサから前記信号転送手段を介して転送される前記第2の命令体系の命令を記憶する命令記憶手段と、

前記命令記憶手段から出力される前記第2の命令体系の命令を、前記第1の命令体系の命令に変換して前記信号転送手段を介して前記プロセッサに与えるための命令トランスレータとを含む、データ処理装置。

【請求項16】 さらに、前記信号転送手段によって前記プロセッサと相互接続される第2の命令メモリを含み、

前記第2の命令メモリは、

前記プロセッサから前記信号転送手段を介して転送される前記第1の命令体系の命令を記憶する命令記憶手段と、

前記プロセッサから前記信号転送手段を介して与えられるアドレス信号に応答して、前記命令記憶手段から出力される前記第1の命令体系の命令を、前記信号転送手段を介して前記プロセッサに与えるための命令読出手段とを含む、請求項15に記載のデータ処理装置。

【請求項17】 さらに、前記トランスレータ付命令メモリから読み出される命令を前記プロセッサに転送するときの前記信号転送手段のウェイト数が、前

記第 2 の命令メモリから読み出される命令を前記プロセッサに転送するときの前記信号転送手段のウェイト数よりも大きくなるように前記信号転送手段による転送を制御するための転送制御手段を含む、請求項 1 6 に記載のデータ処理装置。

【請求項 1 8】 さらに、前記信号転送手段によって前記プロセッサと相互接続される第 3 のトランスレータ付命令メモリを含み、

前記第 3 のトランスレータ付命令メモリは、

前記プロセッサから前記信号転送手段を介して転送される、前記第 2 の命令体系とは異なる第 3 の命令体系の命令を記憶する命令記憶手段と、

前記プロセッサから前記信号転送手段を介して与えられるアドレス信号に応答して、前記命令記憶手段から出力される前記第 3 の命令体系の命令を、前記第 1 の命令体系の命令に変換して前記信号転送手段を介して前記プロセッサに与えるための命令読出手段とを含む、請求項 1 6 に記載のデータ処理装置。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

この発明はあるプロセッサにとっての非ネイティブ命令を、そのプロセッサのネイティブ命令に変換するための命令トランスレータ、そのトランスレータを備えた命令トランスレータ機能付メモリおよびそれらを用いて、非ネイティブ命令を高速に実行することが可能なデータ処理装置に関する。

【0 0 0 2】

【従来の技術】

プロセッサアーキテクチャとそのプロセッサで実行可能な命令体系とは密接な関係を持つ。一方、プロセッサアーキテクチャが進化して命令体系が新しくなると、旧命令体系で実現されたプログラムコードはそのままでは実行不可能となることが通常である。どのようにして旧命令体系で実現されたプログラム資産を有効に承継するかが問題となる。そのため、ある命令体系を持つ新プロセッサで、旧命令体系にしたがって設計された旧プロセッサ用に記述されたプログラムを実行するようにするための手法が多く開発されている。

【0 0 0 3】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するために従来行なわれている代表的な方法に、新プロセッサのハードウェアに旧プロセッサの機能を持たせる方法がある。図1を参照して、そうした方法を実現する従来のデータ処理装置500は、旧プロセッサの命令と新プロセッサの命令との双方をデコードする機能を有する多機能命令デコーダ5およびそれら命令を実行する機能を有する演算部6を備えたプロセッサ1と、プロセッサ1に接続されるバス4と、バス4に接続されるデータメモリ2および命令メモリ3とを含む。

【0004】

命令メモリ3は、旧プロセッサの命令と新プロセッサの命令との双方を保持する。多機能命令デコーダ5は、命令メモリ3から読出され、バス4を介してプロセッサ1に転送された命令をデコードする。このとき、多機能命令デコーダ5は、この命令が新プロセッサ用のものであっても、旧プロセッサ用のものであってもデコードすることができる。デコードされた命令は演算部6によって実行される。データメモリ2は、新プロセッサ用の命令からも、旧プロセッサ用の命令からもアクセスできる。

【0005】

このように新プロセッサのハードウェアに旧プロセッサのハードウェアの機能を持たせる例は、たとえば「IA-32 Application Execution Model in an IA-64 System Environment」(IA-64 Application Developer & Architecture Guide, Chapter 6, May 1999)に詳しく記載されている。

【0006】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するための従来の他の方法として、旧プロセッサ用のソフトウェアを新プロセッサ用のソフトウェアに変換した後に実行する方法、および旧プロセッサ用の命令の動作を新プロセッサ用のソフトウェアでエミュレートする方法がある。この方法は例えば、Tom Thompsonによる「An Alpha in PC Clothing (Digital Equipment's new x86 emulator technology makes an Alpha system a fast x86 clone)」(BYTE, pp. 195-196, February 1996)に詳しく記載されている。

【0007】

ところで、ある命令体系で記述されたプログラムを他の命令体系にしたがって設計されたプロセッサで実行することは、これ以外の場合にも有効である。たとえば、ある命令体系のサブセットを定義してその縮小された命令体系でプログラムを記述すると、プログラムサイズを小さくすることができる。また、J A V A 言語は、仮想的なプロセッサの命令体系を定めてプログラムを記述し、その同一のプログラムを複数のプロセッサでそれぞれのプロセッサの命令体系を用いて実行する仕組みとなっている。このため J A V A 言語で記述されたプログラムは、異なる命令体系の複数種類のプロセッサで共有して実行することができる。

#### 【0008】

プログラムサイズを小さくする目的でサブセットの縮小命令体系を作り、プロセッサの多機能命令デコーダで非縮小命令体系の命令と縮小命令体系の命令との双方の命令をデコードする方法は既に多数提案されている。たとえば James L. Turley による「Thumb Squeezes ARM Code Size (New Core Module Provides Optimized Second Instruction Set)」(Micro Processor Report, Vol. 9, No. 4, pp. 1, 6-9, March 27, 1995) の記載を参照。

#### 【0009】

##### 【発明が解決しようとする課題】

しかし、上述したような従来の手法には、いずれも以下に述べるような問題点がある。

#### 【0010】

複数個の命令体系によって記述されたプログラムを実行する機能をプロセッサのハードウェアに持たせる場合、ハードウェアが複雑になり、かつそのサイズが大きくなる。また、実行すべき命令体系を追加したり変更したりする場合には、ハードウェア全体を設計し直す必要があり、柔軟に対応することが困難である。

#### 【0011】

ソフトウェアによりプログラムを変換する場合には次のような問題がある。プログラム自体を変換する場合には、変換後プログラムを保持するために大容量のメモリを新たに必要とする。その結果、メモリのコストが増大しデータ処理装置のコストも上昇する。また命令の動作を他の命令体系の命令でエミュレートする

場合には、演算結果をエミュレートすることが必要であることはもちろん、プログラムカウンタの値、および必要がある場合にはフラグまでエミュレートしなければならない。その結果、一つの命令の動作を、別の体系の多数の命令で置換することが必要になる。その結果、動作速度が大幅に低下するという問題がある。

#### 【 0 0 1 2 】

この発明はこうした問題を解決するためになされたもので、プロセッサのハードウェア自体は変更せずに、複数個の異なる命令体系の命令からなるプログラムをネイティブ命令を用いて高速に実行可能なデータ処理装置であって、大容量のメモリを必要としないデータ処理装置、そのための命令トランスレータおよび命令トランスレータ機能付メモリを提供することを目的とする。

#### 【 0 0 1 3 】

##### 【課題を解決するための手段】

請求項 1 に記載の発明にかかる命令トランスレータは、第 1 の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、第 1 の命令体系と異なる第 2 の命令体系の命令を記憶する命令メモリとともに用いられ、第 1 の命令体系の命令に変換してプロセッサに与えるための命令トランスレータであって、プロセッサが実行すべき命令の、命令メモリにおけるアドレスを受けて命令メモリから対応する命令を読み出し、読み出された第 2 の命令体系の命令を第 1 の命令体系の命令に変換するための変換手段と、変換手段により変換された後の第 1 の命令体系の命令を、命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、プロセッサが実行すべき命令のアドレスを受けて保持手段を探索し、保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、変換手段の出力する命令と、保持手段に保持されていた、対応の第 1 の命令体系の命令とを選択的にプロセッサに出力するための選択手段とを含む。

#### 【 0 0 1 4 】

請求項 1 に記載の発明によれば、命令トランスレータを追加することにより、プロセッサ本体の構成を変更することなく非ネイティブ命令をネイティブ命令に変換してプロセッサで実行することができる。また、第 1 の命令体系に変換され

た命令を一時的に保持手段に保持し、次に当該命令の読出が命令メモリから行なわれるときは、保持手段に保持されていた変換済みの第1の命令体系の命令を出力することができるため、命令メモリからの読出処理と変換処理とを省略することができ、高速に変換後の命令を出力できる。

## 【 0 0 1 5 】

請求項2に記載の発明にかかる命令トランスレータは、請求項1に記載の発明の構成に加えて、第2の命令体系は可変長命令体系であり、変換手段は、命令メモリから読出された第2の命令体系の命令を、読出された第2の命令体系の命令の命令長に依存した数の第1の命令体系の命令に変換するための可変長命令変換手段を含む。

## 【 0 0 1 6 】

請求項2に記載の発明によれば、請求項1に記載の発明の作用に加えて、非ネイティブ命令の長さを変換後のネイティブ命令の個数でエミュレートするため、非ネイティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がない。

## 【 0 0 1 7 】

請求項3に記載の発明にかかる命令トランスレータは、請求項2に記載の発明の構成に加えて、可変長命令変換手段は、命令メモリから読出された第2の命令体系の命令を、読出された第2の命令体系の命令の命令長に依存し、かつそれよりも長い長さの第1の命令体系の命令に変換するための第1の手段を含む。

## 【 0 0 1 8 】

請求項3に記載の発明によれば、請求項2に記載の発明の作用に加えて、非ネイティブ命令の長さを、その非ネイティブ命令の長さに依存した変換後のネイティブ命令の長さによりエミュレートするので、非ネイティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がない。かつ変換前の非ネイティブ命令の平均長が短くなるため、プログラムサイズが小さくなりプログラムを保存するためのメモリが小さくてすむ。

## 【 0 0 1 9 】

請求項4に記載の発明にかかる命令トランスレータは、請求項3に記載の発明

の構成に加えて、第 1 の命令体系の各命令は 1 または複数個のサブ命令を含み、第 1 の手段が変換する第 1 の命令体系の命令に含まれるサブ命令の個数は、読み出された第 2 の命令体系の命令の長さに依存する。

## 【 0 0 2 0 】

請求項 4 に記載の発明によれば、請求項 3 に記載の発明の作用に加えて、変換後の第 1 の命令体系の命令に複数個のサブ命令を含ませることにより、複数個の処理を指定することができる。そのため非ネイティブ命令である第 2 の命令体系の命令のエミュレートを容易に行なうことができる。さらに、非ネイティブ命令の長さをサブ命令の個数に依存する第 1 の命令体系のネイティブ命令の長さでエミュレートできる。

## 【 0 0 2 1 】

請求項 5 に記載の発明にかかる命令トランスレータは、請求項 1 に記載の発明の構成に加えて、変換手段は、命令メモリから読出された第 2 の命令体系の複数個の命令を、1 つの第 1 の命令体系の命令に変換するための複数命令変換手段を含む。

## 【 0 0 2 2 】

請求項 5 に記載の発明によれば、請求項 1 に記載の発明の作用に加えて、複数の非ネイティブ命令を 1 つのネイティブ命令に一度に変換することができるので、命令の変換効率がよい。

## 【 0 0 2 3 】

請求項 6 に記載の発明にかかる命令トランスレータは、請求項 1 に記載の発明の構成に加えて、第 1 の命令体系の各命令は、1 または複数個のサブ命令を含むことが可能であり、変換手段は、命令メモリから読出された第 2 の命令体系の複数個の命令を、複数個の命令の数に依存した数のサブ命令を含む第 1 の命令体系の命令に変換するための手段を含む。

## 【 0 0 2 4 】

請求項 6 に記載の発明によれば、請求項 1 に記載の発明の作用に加えて、複数個のサブ命令によって複数個の非ネイティブ命令をエミュレートするので命令の変換が容易であり、かつ、ネイティブ命令の長さはそこに含まれるサブ命令の個

数に依存するので、非ネイティブ命令のプログラムカウンタ値を変換後のネイティブ命令の長さでエミュレートすることができる。

【0025】

請求項7に記載の発明にかかる命令トランスレータは、請求項6に記載の発明の構成に加えて、変換後の第1の命令体系の命令に含まれるサブ命令の個数が、複数個の命令の数と等しい。

【0026】

請求項7に記載の発明によれば、請求項6に記載の発明の作用に加えて、変換前の非ネイティブ命令の個数と変換後のサブ命令の個数とが等しいので、非ネイティブ命令とネイティブ命令のサブ命令とを対照させることにより、変換が容易に行える。

【0027】

請求項8に記載の発明にかかる命令トランスレータは、第1の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、第1の命令体系と異なる第2の命令体系の命令を記憶する命令メモリとともに用いられ、第1の命令体系の命令に変換してプロセッサに与えるための命令トランスレータであって、プロセッサが実行すべき命令の、命令メモリにおけるアドレスを受けて命令メモリから対応する命令を読み出し、読み出された第2の命令体系の命令を第1の命令体系の1または複数個の命令に変換する変換手段と、変換手段により変換された後の第1の命令体系の命令を、命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、プロセッサが実行すべき命令のアドレスを受けて保持手段を探索し、保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、変換手段の出力する命令と、保持手段に保持されていた、対応の、第1の命令体系の命令とを選択的にプロセッサに出力するための選択手段と、保持手段に保持されている命令を第1の条件および第2の条件のいずれかで無効化可能なエントリとして保持するよう保持手段を制御するための保持制御手段とを含む。

【0028】

請求項8に記載の発明によれば、1つの非ネイティブ命令を1または複数個の

ネイティブ命令に変換して保持手段に保持することにより、次に同じ非ネイティブ命令の読出が行なわれるときには保持手段から変換後のネイティブ命令を高速に出力することができる。さらに、保持手段に保持されている命令の無効化に対して第1の条件および第2の条件のいずれかとすることができるので、複数のネイティブ命令の同時無効化など、無効化のための条件が複雑な場合でも容易に対処できる。

## 【 0 0 2 9 】

請求項9に記載の発明にかかる命令トランスレータは、請求項8に記載の発明の構成に加えて、第1の条件は保持手段による所定のアルゴリズムに基づくハードウェア制御による保持制御の条件であり、第2の条件は保持手段の外部からの明示的な無効化指示があったという条件である。

## 【 0 0 3 0 】

請求項9に記載の発明によれば、請求項8に記載の発明の作用に加えて、ハードウェア制御により保持手段の保持内容を維持することに加えて、外部から明示的に保持手段の内容を無効化することが可能であり、ソフトウェアの責任において安全に保持手段の保持内容を維持することができる。

## 【 0 0 3 1 】

請求項10に記載の発明にかかる命令トランスレータは、請求項8に記載の発明の構成に加えて、保持制御手段は第2の条件で無効化が可能なエントリを無効化することなしに新たな命令を保持手段に保持できないときにアサートされる信号を出力する。

## 【 0 0 3 2 】

請求項10に記載の発明によれば、請求項8に記載の発明の作用に加えて、新たな命令を保持手段に保持できないときには、外部に信号がアサートされる。この信号に応答して、無効化しても安全なエントリを無効化することをソフトウェア処理によって明示的に指示して、新たな命令を保持可能とすることができる。

## 【 0 0 3 3 】

請求項11に記載の発明にかかるトランスレータ付命令メモリは、第1の命令体系の命令をネイティブ命令として動作するプロセッサとともに用いられるトラ

ンスレータ付命令メモリであって、第2の命令体系の命令を記憶する命令記憶手段と、命令記憶手段から出力される第2の命令体系の命令を、第1の命令体系の命令に変換してプロセッサに与えるための命令トランスレータとを含む。

## 【 0 0 3 4 】

請求項11に記載の発明によれば、命令記憶手段に第2の命令体系の命令を記憶しておき、これを命令トランスレータで第1の命令体系の命令に変換してプロセッサに与えることができる。プロセッサの変更なしに、非ネイティブ命令により記述されたプログラムを実行できる。

## 【 0 0 3 5 】

請求項12に記載の発明にかかるトランスレータ付命令メモリは、請求項11に記載の発明の構成に加えて、命令トランスレータは、命令記憶手段から読み出すべき命令のアドレスに基づいて、第2の命令体系の命令を、第1の命令体系の命令に変換する処理と、第2の命令体系の命令をそのまま出力する処理とのいずれかを選択的に実行するための手段を含む。

## 【 0 0 3 6 】

請求項12に記載の発明によれば、請求項11に記載の発明の作用に加えて、第2の命令体系の命令を第1の命令体系の命令に変換して読出すことが可能であることに加え、第2の命令体系の命令をそのまま読出すことも可能なので、第2の命令体系で記述されたプログラムを他のメモリに転送したり、その内容を解析したりすることが可能になる。

## 【 0 0 3 7 】

請求項13に記載の発明にかかるトランスレータ付命令メモリは、請求項11に記載の発明の構成に加えて、命令記憶手段からの読出時にアドレス変換を行なうためのアドレス変換手段をさらに含む。

## 【 0 0 3 8 】

請求項13に記載の発明によれば、請求項11に記載の発明の作用に加えて、命令変換時と、それ以外のときとで、トランスレータ付命令メモリに関して異なるメモリマップを使用することができる。

## 【 0 0 3 9 】

請求項 1 4 に記載の発明にかかるトランスレータ付命令メモリは、請求項 1 1 に記載の発明の構成に加えて、命令トランスレータは、プロセッサが実行すべき命令の、トランスレータ付命令メモリにおけるアドレスを受けてトランスレータ付命令メモリから対応する命令を読み出し、読み出された第 2 の命令体系の命令を第 1 の命令体系の命令に変換する変換手段と、変換手段により変換された後の第 1 の命令体系の命令を、トランスレータ付命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、プロセッサが実行すべき命令のアドレスを受けて保持手段を探索し、保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、変換手段の出力する命令と、保持手段に保持されていた、対応の第 1 の命令体系の命令とを選択的にプロセッサに出力するための選択手段とを含む。

## 【 0 0 4 0 】

請求項 1 4 に記載の発明によれば請求項 1 1 に記載の発明の作用に加えて、第 1 の命令体系の命令に変換された命令は保持手段に保持され、次に読出が行なわれるときにその命令が保持手段に保持されていれば、あらためて命令の変換を行なうことなく、保持手段から対応の変換後の命令が出力されるので、命令記憶手段へのアクセスと、変換とに要する時間が不要となり、高速に変換後の命令を出力することができる。

## 【 0 0 4 1 】

請求項 1 5 に記載の発明にかかるデータ処理装置は、第 1 の命令体系の命令をネイティブ命令として動作するプロセッサと、プロセッサが接続される信号転送手段と、信号転送手段を介してプロセッサと相互接続されるトランスレータ付命令メモリとを含み、トランスレータ付命令メモリは、プロセッサから信号転送手段を介して転送される第 2 の命令体系の命令を記憶する命令記憶手段と、命令記憶手段から出力される第 2 の命令体系の命令を、第 1 の命令体系の命令に変換して信号転送手段を介してプロセッサに与えるための命令トランスレータとを含む。

## 【 0 0 4 2 】

請求項 1 5 に記載の発明によれば、プロセッサにとって非ネイティブ命令であ

る第2の命令体系の命令を、プロセッサのネイティブ命令である第1の命令体系の命令に変換する命令トランスレータが設けられているので、プロセッサの構成を変更することなく、第2の命令体系の命令で記述されたプログラムをこのプロセッサで実行することが可能となる。

【0043】

請求項16に記載の発明にかかるデータ処理装置は、請求項15に記載の発明の構成に加えて、さらに、信号転送手段によってプロセッサと相互接続される第2の命令メモリを含み、第2の命令メモリは、プロセッサから信号転送手段を介して転送される第1の命令体系の命令を記憶する命令記憶手段と、プロセッサから信号転送手段を介して与えられるアドレス信号に応答して、命令記憶手段から出力される第1の命令体系の命令を、信号転送手段を介してプロセッサに与えるための命令読出手段とを含む。

【0044】

請求項16に記載の発明によれば、請求項15に記載の発明の作用に加えて、第1の命令体系の命令で記述されたプログラムも第2の命令メモリからプロセッサに転送して実行できるので、プロセッサ本体の変更なしに非ネイティブ命令もネイティブ命令も区別なくプロセッサでデコードし実行することができる。

【0045】

請求項17に記載の発明にかかるデータ処理装置は、請求項16に記載の発明の構成に加えて、さらに、トランスレータ付命令メモリから読み出される命令をプロセッサに転送するときの信号転送手段のウェイト数が、第2の命令メモリから読み出される命令をプロセッサに転送するときの信号転送手段のウェイト数よりも多くなるように信号転送手段による転送を制御するための転送制御手段を含む。

【0046】

請求項17に記載の発明によれば、請求項16に記載の発明の作用に加えて、トランスレータ付命令メモリから読み出される命令をプロセッサに転送するときのウェイト数が多いので、その間に非ネイティブ命令からネイティブ命令への変換を行なうことができ、プロセッサでは読出対象となった命令が非ネイティブ命

令の場合もネイティブ命令の場合も区別なくフェッチしてデコードすることができる。

【0047】

請求項18に記載の発明にかかるデータ処理装置は、請求項16に記載の発明の構成に加えて、さらに、信号転送手段によってプロセッサと相互接続される第3のトランスレータ付命令メモリを含み、第3のトランスレータ付命令メモリは、プロセッサから信号転送手段を介して転送される、第2の命令体系とは異なる第3の命令体系の命令を記憶する命令記憶手段と、プロセッサから信号転送手段を介して与えられるアドレス信号に応答して、命令記憶手段から出力される第3の命令体系の命令を、第1の命令体系の命令に変換して信号転送手段を介してプロセッサに与えるための命令読出手段とを含む。

【0048】

請求項18に記載の発明によれば、請求項16に記載の発明の作用に加えて、トランスレータ付メモリおよび第3のトランスレータ付命令メモリは、それぞれ異なる種類の非ネイティブ命令をネイティブ命令に変換して信号転送手段に出力するので、プロセッサはどちらの非ネイティブ命令が変換された命令かを区別することなく、非ネイティブ命令で記述されたプログラムをネイティブ命令を用いて実行できる。

【0049】

【発明の実施の形態】

図2を参照して、この発明の実施の形態のデータ処理装置は、プロセッサ10と、プロセッサ10に接続されたバス40と、バス40、READY信号線50およびプロセッサ10に接続されたバス制御回路20と、いずれもREADY信号線50およびバス40に接続された、トランスレータ14を有する圧縮命令用のトランスレータ付メモリ24、トランスレータ15を有するJAVA命令用のトランスレータ付メモリ25およびトランスレータ16を有する非ネイティブ命令X用のトランスレータ付メモリ26と、いずれもバス40に接続された、ネイティブ命令用RAM21、データ用メモリ22および、ネイティブ命令と、圧縮命令と、JAVA命令と、非ネイティブ命令Xと、データとを格納するROM2

3とを含む。

#### 【0050】

バス制御回路20は、プロセッサ10からバス40に出力されたアドレスをデコードしてネイティブ命令用RAM21、データ用メモリ22、ROM23、圧縮命令用のトランスレータ付メモリ24、JAVA命令用のトランスレータ付メモリ25および非ネイティブ命令X用のトランスレータ付メモリ26に対してこれらのいずれかをアクティベートするチップセレクト信号CSを出力する。バス制御回路20はまた、トランスレータ14~16に対して、これらの命令変換機能を制御する変換機能イネーブル信号TEを出力してトランスレータ14~トランスレータ16に与える。またバス制御回路20は、トランスレータ14~16からREADY信号線50を介して与えられるREADY信号を受け、バスサイクルの終了を示すDC信号51をプロセッサ10に与える。

#### 【0051】

図3を参照して、プロセッサ10は、コア100と、命令キャッシュ101と、データキャッシュ102と、バス40およびDC信号51に接続されたバスインターフェイス部103と、これらを互いに接続する命令アドレスバス104および命令バス105と、コア100、命令キャッシュ101およびバスインターフェイス部103を相互に接続してアドレスおよびデータを送信するためのデータアドレスバス106およびデータバス107とを含む。

#### 【0052】

コア100は、VLIW (Very Long Instruction Word) 型命令体系を有するプロセッサである。コア100は、命令バス105から入力されたVLIW命令をデコードするための命令デコーダ110と、命令デコーダ110によってデコードされた命令を実行するためのメモリ演算部130および整数演算部140と、メモリ演算部130および整数演算部140に複数のバスで接続されたレジスタファイル120とを含む。

#### 【0053】

命令デコーダ110は二つのサブ命令デコーダ111および112を含む。

メモリ演算部130は、アドレス演算器131、PC演算器132、シフト1

33およびALU134などの演算器を含む。メモリ演算部130は、サブ命令デコーダ111の出力にしたがい、メモリアクセス命令、PC制御命令、整数演算命令などを実行するためのものである。整数演算部140は、シフタ141、ALU142、乗算器143およびアキュムレータ144を含む。整数演算部140は、サブ命令デコーダ112の出力にしたがい整数演算命令を実行するためのものである。メモリ演算部130および整数演算部140は、2つのサブ命令を並列に実行する場合と、それぞれ独立に1つのサブ命令を実行する場合とがある。

#### 【0054】

図4を参照して、プロセッサ10が有するレジスタファイル120は、64本の汎用レジスタであるレジスタ150～152、162、163a、および163bを含む。プロセッサ10はさらに、制御レジスタ170～180を含む。また、図3に示すアキュムレータ144はアキュムレータ144aおよびアキュムレータ144bを含む。

#### 【0055】

レジスタ150は常にゼロを保持するレジスタである。レジスタ162は非割込処理中のスタックトップのデータを保持するためのものである。レジスタ163bは非割込処理中のスタックポインタでスタックトップのすぐ下のデータのアドレスを保持するためのものである。レジスタ163aおよび163bは、PSW（プロセッサステータスワード）である制御レジスタ170中にあるモードビットで切り替わり、割込処理中はレジスタ163aが使用され、非割込処理中はレジスタ163bが使用される。

#### 【0056】

制御レジスタ170～180は、それぞれ所定の要素のための専用のレジスタである。たとえば制御レジスタ170はPSWであって、演算により変化するフラグ、割込処理処理中か否か、割込マスク中か否か、デバック中か否かなど、プロセッサ10の動作モードを示すモードビットを含む。制御レジスタ172はプログラムカウンタ（PC）であり、現在実行中の命令のアドレスを示す。制御レジスタ171および173は、割込受付時、例外発生時、トラップ発生時にそれ

ぞれ制御レジスタ170および172の値をコピーして保持するためのものである。

#### 【0057】

アキュムレータ144aおよび144bは、乗算結果、積和演算結果を保持するためのものである。アキュムレータ144aおよび144bはそれぞれ、汎用レジスタの2倍のビット長である64ビットのデータを保持することができる。

#### 【0058】

図5を参照して、制御レジスタ170が保持するPSWは32ビットであって、割込処理中か非割込処理中かを示すモードビットであるSMビット170aと、割込許可中か割込禁止中かを示すIEビット170bと、命令の実行条件を制御するF0ビット170cおよびF1ビット170dとを含む。この他に制御レジスタ170は、RPビット、MDビット、F2～F7の各ビットを含む。これらの意味については図5に示す通りである。

#### 【0059】

図6を参照して、プロセッサ10は、命令を以下のようにしてパイプライン処理する。プロセッサ10は、メモリ演算部130と整数演算部140とで行なわれるサブ命令をそれぞれ実行するためのMUパイプ139およびIUパイプ149を含む。これらパイプはいずれも、命令フェッチステージ191、デコードおよびアドレス計算ステージ192、演算およびメモリアクセスステージ193およびライトバックステージ194からなる。

#### 【0060】

命令フェッチステージ191は、命令をフェッチして命令デコーダ110中の命令レジスタ113に保持するステージである。デコードおよびアドレス計算ステージ192では、この命令がサブ命令デコーダ111、112でデコードされ、同時にレジスタファイル120がアクセスされてオペランドおよびPCのアドレス計算が行なわれる。演算およびメモリアクセスステージ193では、整数演算およびデータメモリアクセス処理が行なわれる。ライトバックステージ194では、演算結果およびメモリからフェッチされたデータがレジスタファイル120に再び書込まれる。

## 【0061】

図7を参照して、プロセッサ10の命令200は2ウェイのVLW型命令であり、図示されるようなフォーマットを有する。すなわち、命令200は、各サブ命令の実行順序と長いサブ命令を定義するFMフィールド201a、201bと、サブ命令を格納するLコンテナ205およびRコンテナ206と、各サブ命令の実行条件を指定する条件フィールド203および204とを含む。

## 【0062】

条件フィールド203および204は、制御レジスタ170であるPSW中のフラグF0およびF1（F0ビット170cおよびF1ビット170d）の値に依存した条件を指定する。たとえば条件フィールド203が「000」のとき、Lコンテナ205に含まれるサブ命令は無条件に実行される。条件フィールド204が「101」のとき、Rコンテナ206に含まれるサブ命令はF0=1かつF1=1のとき実行され、フラグF0およびF1（F0ビット170cおよびF1ビット170d）がそれ以外の値をとる場合は無効化される。

## 【0063】

FMフィールド201aおよび201bは、Lコンテナ205とRコンテナ206とに含まれるサブ命令を実行する場合の実行動作を指定する。実行動作としては4つある。1番目は、Lコンテナ205とRコンテナ206とに含まれるサブ命令を並列に実行する動作である。2番目は、Lコンテナ205のサブ命令をまず実行して次にRコンテナ206のサブ命令を実行する動作である。3番目は2番目の動作の逆であり、Rコンテナ206のサブ命令をまず実行して次にLコンテナ205のサブ命令を実行する動作である。4番目はLコンテナ205とRコンテナ206とに分割して保持された1つの長いサブ命令を実行する動作である。すなわち、FMフィールド201aおよびFMフィールド201bの値によって、上述した4つの動作のいずれかが選択される。

## 【0064】

図8を参照して、Lコンテナ205およびRコンテナ206に保持されるサブ命令は以下のようなフォーマットを有する。サブ命令は28ビット長の短い命令と54ビット長の長いサブ命令とに分類される。短い命令は、フォーマット21

1～217に示される7種類のフォーマットを有する。短い命令フォーマットの概略をいえば、ビット位置0～9で演算の種類が示され、ビット位置10～27で最大3つのオペランドが指定される。長いサブ命令はフォーマット218に示されるようにビット位置0～9で演算の種類が示され、ビット位置10～53で32ビット長の即値データを含む最大3つのオペランドが指定される。なお、長いサブ命令の32ビットの即値はVLIW命令ビット位置26～31、36～43、および46～63に保持される。

#### 【0065】

フォーマット211は、メモリアクセス演算（ロード／ストア演算）を行なうサブ命令のフォーマットである。フォーマット212は汎用レジスタに保持されたオペランド間の演算（OP演算）を行なうサブ命令のフォーマットである。フォーマット213～217は分岐演算を行なうサブ命令のフォーマットである。長いサブ命令のフォーマット218は、上記した3種類の演算全てに共通で 사용되는。

#### 【0066】

サブ命令を図6に示すようにプロセッサ10でパイプライン処理する場合には、OP演算、ロード／ストア演算、分岐演算のサブ命令はそれぞれ図9に示す4段のパイプライン221～223によって示されるように4段のパイプラインステージで実行される。

#### 【0067】

FMフィールド201aおよび201bによりサブ命令の実行順序が指定されたときには、サブ命令はMUパイプ139およびIUパイプ149によって図10に示すようにパイプライン処理される。ここで、ストールステージ234～236は、FMフィールド201aおよび201bの値にしたがってサブ命令に順序をつけて実行する場合に、一方のサブ命令の実行を遅延させるために挿入されるパイプラインステージである。

#### 【0068】

次に、プロセッサ10に対して定義されたサブ命令の一覧を示す。この一覧において、各サブ命令のニーモニックを大文字で、その処理内容を各ニーモニック

の右側に、それぞれ示す。

【0069】

ロード／ストア命令

|        |                          |
|--------|--------------------------|
| LDB    | 符号拡張してレジスタに1バイトをロード      |
| LDBU   | ゼロ拡張してレジスタに1バイトをロード      |
| LDH    | 符号拡張してレジスタに半ワードをロード      |
| LDHH   | レジスタ上位に半ワードをロード          |
| LDHU   | ゼロ拡張してレジスタに半ワードをロード      |
| LDW    | レジスタに1ワードをロード            |
| LD2W   | 2つのレジスタに2ワードをロード         |
| LD4BH  | 符号拡張して4つの半ワードレジスタに4つのバイト |
| をロード   |                          |
| LD4BHU | ゼロ拡張して4つの半ワードレジスタに4つのバイト |
| をロード   |                          |
| LD2H   | 2つのレジスタに2つの半ワードをロード      |
| STB    | レジスタから1バイトをストア           |
| STH    | レジスタから半ワードをストア           |
| STHH   | レジスタ上位から半ワードをストア         |
| STW    | レジスタから1ワードをストア           |
| ST2W   | レジスタから2ワードをストア           |
| ST4HB  | 4つの半ワードレジスタから4つのバイトをストア  |
| ST2H   | 2つのレジスタから2半ワードをストア       |
| MODDEC | 5ビット即値によってレジスタ値をデクリメント   |
| MODINC | 5ビット即値によってレジスタ値をインクリメント  |

転送命令

|        |                   |
|--------|-------------------|
| MVFSYS | 制御レジスタから汎用レジスタに転送 |
| MVTSYS | 汎用レジスタから制御レジスタに転送 |
| MVFACC | アキュムレータから1ワードを転送  |

MVTACC                      2 つの汎用レジスタから 2 ワードをアキュムレータに  
転送

## 比較命令

CMPcc

比較

cc= EQ(000), NE(001), GT(010), GE(011), LT(100),  
LE(101), PS-both positive(110), NG-both neg

ative(111)

CMPUcc

符号なし比較

cc=GT(010), GE(011), LT(100), LE(101)

## 算術演算命令

ABS

絶対値

ADD

加算

ADDC

キャリー付加算

ADDHppp

半ワード加算

ppp=LLL(000), LLH(001), LHL(010), LHH(011),

HLL(100), HLH(101), HHL(110), HHH(111)

ADDS

第 3 オペランドの符号をレジスタ Rb に加算

ADDS2H

2 つの半ワードに符号を加算

ADD2H

2 対の半ワードを加算

AVG

正の無限大方向の丸めによる平均

AVG2H

正の無限大方向の丸めにより 2 対の半ワードを平均

JOINpp

2 つの半ワードをジョイン

pp=LL(00), LH(01), HL(10), HH(11)

SUB

減算

SUBB

ボロー付の減算

SUBHppp

半ワードの減算

ppp=LLL(000), LLH(001), LHL(010), LHH(011),

HLL(100),HLH(101),HHL(110),HHH(111)

SUB2H 2 対の半ワードの減算

### 論理演算命令

|       |             |
|-------|-------------|
| AND   | 論理AND       |
| OR    | 論理OR        |
| NOT   | 論理NOT       |
| XOR   | 排他的論理OR     |
| ANDFG | フラグの論理AND   |
| ORFG  | フラグの論理OR    |
| NOTFG | フラグの論理NOT   |
| XORFG | フラグの排他的論理OR |

### シフト演算命令

|       |                 |             |
|-------|-----------------|-------------|
| SRA   | 右に算術シフト         |             |
| SRAHp | 右に半ワードを算術シフト    | p=L(0),H(1) |
| SRA2H | 2つの半ワードを右に算術シフト |             |
| SRC   | レジスタを接続して右にシフト  |             |
| SRL   | 右に論理シフト         |             |
| SRLHp | 半ワードを右に論理シフト    | p=L(0),H(1) |
| SRL2H | 2つの半ワードを右に論理シフト |             |
| ROT   | 右ローテイト          |             |
| ROT2H | 2つの半ワードを右ローテイト  |             |

### ビット演算命令

|      |         |
|------|---------|
| BCLR | ビットをクリア |
| BNOT | ビットを反転  |
| BSET | ビットをセット |
| BTST | ビットをテスト |

## 分岐命令

|        |                    |
|--------|--------------------|
| BRA    | 分岐                 |
| BRATZR | ゼロなら分岐             |
| BRATNZ | ゼロでなければ分岐          |
| BSR    | サブルーチンに分岐          |
| BSRTZR | ゼロならサブルーチンに分岐      |
| BSRTNZ | ゼロでなければサブルーチンに分岐   |
| DBRA   | 遅延分岐               |
| DBRAI  | 即値付遅延分岐            |
| DBSR   | サブルーチンへの遅延分岐       |
| DBSRI  | サブルーチンへの即値付遅延分岐    |
| DJMP   | 遅延ジャンプ             |
| DJMPI  | 即値付遅延分岐            |
| DJSR   | サブルーチンへの遅延ジャンプ     |
| DJSRI  | サブルーチンへの即値付遅延ジャンプ  |
| JMP    | ジャンプ               |
| JMPTZR | ゼロならジャンプ           |
| JMPTNZ | ゼロでなければジャンプ        |
| JSR    | サブルーチンに分岐          |
| JSRTZR | ゼロならサブルーチンにジャンプ    |
| JSRTNZ | ゼロでなければサブルーチンにジャンプ |
| NOP    | ノーオペレーション          |

## OS関連命令

|      |                   |
|------|-------------------|
| TRAP | トラップ              |
| REIT | 例外、割込およびトラップからの復帰 |

## DSP算術演算命令

|         |                                   |         |
|---------|-----------------------------------|---------|
| MUL     | 乗算                                |         |
| MULX    | 拡張精度の乗算                           |         |
| MULXS   | 拡張精度での乗算および左へのシフト                 |         |
| MULX2H  | 拡張精度で 2 対の半ワードを乗算                 |         |
| MULHXpp | 拡張精度で 2 つの半ワードを乗算                 |         |
|         | pp=LL(00), LH(01), HL(10), HH(11) |         |
| MUL2H   | 2 対の半ワードを乗算                       |         |
| MACd    | 乗算および加算                           | (d=0,1) |
| MACSd   | 乗算、左 1 ビットシフト、加算                  | (d=0,1) |
| MSUBd   | 乗算および減算                           | (d=0,1) |
| MSUBSd  | 乗算、左 1 ビットシフト、減算                  | (d=0,1) |
| SAT     | サチュレート                            |         |
| SATHH   | ワードオペランドを上位半ワードにサチュレート            |         |
| SAHL    | ワードオペランドを下位半ワードにサチュレート            |         |
| SATZ    | 正数にサチュレート                         |         |
| SATZ2H  | 2 つの半ワードを正数にサチュレート                |         |
| SAT2H   | 2 つの半ワードオペランドをサチュレート              |         |

#### 繰返し命令

|         |                  |
|---------|------------------|
| REPEAT  | 1 ブロックの命令を繰返す    |
| REPEATI | 1 ブロックの命令を即値付繰返す |

#### デバッグサポート命令

|     |                    |
|-----|--------------------|
| DBT | デバッグトラップ           |
| RTD | デバッグ割込およびトラップからの復帰 |

図 1 1 に、図 2 に示すプロセッサ 1 0 がバス 4 0 を介してネイティブ命令用 R  
AM 2 1、データ用メモリ 2 2、ROM 2 3、圧縮命令用のトランスレータ付メ  
モリ 2 4、J A V A 命令用のトランスレータ付メモリ 2 5 および非ネイティブ命

令X用のトランスレータ付メモリ26をアクセスする場合のアドレスマップを示す。図11に示されるように、ネイティブ命令用RAM21、データ用メモリ22およびROM23はそれぞれ、ネイティブ命令用アドレス領域121、データメモリ用アドレス領域122およびROM用アドレス領域123にマップされる。圧縮命令用のトランスレータ付メモリ24、JAVA命令用のトランスレータ付メモリ25および非ネイティブ命令X用のトランスレータ付メモリ26はそれぞれ、命令コードの変換を行わずにリードライトする場合には圧縮命令（直結）用アドレス領域124a、JAVA命令用（直結）アドレス領域125aおよび非ネイティブ命令X（直結）用アドレス領域126aにマップされる。圧縮命令用のトランスレータ付メモリ24、JAVA命令用のトランスレータ付メモリ25および非ネイティブ命令X用のトランスレータ付メモリ26は、命令コードの変換を伴ってリードする場合にはそれぞれ圧縮命令用領域（トランスレータ経由）124b、JAVA命令（トランスレータ経由）用アドレス領域125b、およびアドレス領域126bにそれぞれマップされる。

#### 【0070】

図12を参照して、圧縮命令用のトランスレータ付メモリ24は、トランスレータ14と圧縮命令用RAM245とを含む。トランスレータ14はトランスレーション回路243と、入力された値を1ビット右シフトして1/2の値としてアドレス線253に出力するためのアドレス変換器241と、TE信号線250上の変換機能イネーブル信号TEが命令の伸張を指示している場合に、アドレス線253を選択して値が1/2にされたアドレスを選択し、それ以外の場合にはアドレス線252の値を選択して、圧縮命令用RAM245とトランスレーション回路243とに与えるためのMUX242と、圧縮命令用RAM245ヘデータ線259から命令を書込むか、読出すかを指定するR/W信号251とアドレス線252上のアドレスとに基づいて、圧縮命令用RAM245に対して書込みがあると、トランスレーション回路243中にあるキャッシュメモリの、アドレス線252上のアドレスに対応するエントリを無効化することを指示する制御信号257aを出力してトランスレーション回路243に与えるための書込検出回路246とを含む。制御信号257aがアサートされると、トランスレーション

回路243はそのときのアドレス線254上のアドレスを見て、対応するキャッシュのエントリを無効化する。これにより、圧縮命令用RAM245の古い内容に基づく古い変換結果がキャッシュから読出されることを防ぐ。また、アドレス変換器241により、圧縮命令用のトランスレータ付メモリ24へのアクセスアドレスが変更される。

#### 【0071】

圧縮命令用のトランスレータ付メモリ24はさらに、圧縮命令用のトランスレータ付メモリ24から命令コードを読出す場合に、圧縮命令用RAM245から命令コード出力線257に出力された命令コードをそのままデータ線259へ出力するか、それをトランスレーション回路243で伸張した後の命令コードを伸張後命令コード出力線258からデータ線259へ出力するかのいずれか一方を選択するためのMUX244を含む。トランスレーション回路243はまた、伸張後の命令コードを出力するタイミングをバス制御回路20へのREADY信号線50に与えるための、トライステート信号であるREADY信号をREADY信号線256上に出力する。

#### 【0072】

プロセッサ10から見ると、圧縮命令用のトランスレータ付メモリ24のアドレスは圧縮命令（直結）用アドレス領域124aと圧縮命令用領域（トランスレータ経由）124bとからなる。圧縮命令（直結）用アドレス領域124aで圧縮命令用のトランスレータ付メモリ24をアクセスする場合、64kBの空間の全バイト位置に有効なメモリがある。一方、圧縮命令用領域（トランスレータ経由）124bで圧縮命令用のトランスレータ付メモリ24をアクセスする場合、128kBの空間のアラインされた各8バイトの上位4バイトには有効なメモリがあるが、下位4バイトには有効なメモリは存在しない。圧縮命令用のトランスレータ付メモリ24では、もともと8バイトのVLW命令の中の2つのサブ命令を、アドレスを変えずにそれぞれ2バイトに圧縮して保存しているためである。プロセッサ10がたとえばアドレスH' 20020100から圧縮されたVLW命令をフェッチして実行するとき、圧縮命令用のトランスレータ付メモリ24からは2バイトの命令2つが1つの8バイトのVLW命令に伸張されて出

力され、プロセッサ 1 0 では P C 値を 8 バイト進めてアドレス H' 2 0 0 2 0 1 0 8 から次の命令をフェッチして実行する。

#### 【 0 0 7 3 】

図 1 3 を参照して、トランスレーション回路 2 4 3 は、アドレス線 2 5 4 から与えられるアドレスに対応する伸張後の命令を保持している場合には伸張後の V L I W 命令を出力線 3 5 9 に出力し、あわせてキャッシュヒット信号線 3 5 7 上のキャッシュヒット信号をアサートするキャッシュメモリ 3 5 4 と、命令コード出力線 2 5 7 から入力された 2 つの命令を 8 バイトの V L I W 命令に伸張して出力線 3 6 0 に出力し、あわせて命令の伸張が終了したタイミングを示す信号をタイミング信号線 3 5 8 に出力するための命令コード伸張部 3 5 0 と、キャッシュメモリ 3 5 4 からの出力線 3 5 9 と命令コード伸張部 3 5 0 からの出力線 3 6 0 との一方を選択して出力するための MUX 3 5 6 と、キャッシュメモリ 3 5 4 からのキャッシュヒット信号線 3 5 7 と命令コード伸張部 3 5 0 からのタイミング信号線 3 5 8 との論理和を出力するための O R 回路 3 5 5 とを含む。命令コード伸張部 3 5 0 は、アドレス線 2 5 4 上のアドレスに基づいて、命令コード出力線 2 5 7 から入力された 2 つの圧縮された命令の一方を選択して出力するための MUX 3 5 1 と、MUX 3 5 1 の出力する命令を伸張する伸張器 3 5 2 とを含む。

#### 【 0 0 7 4 】

アドレス線 2 5 4 から入力されたアドレスはキャッシュメモリ 3 5 4 と命令コード伸張部 3 5 0 とへ入力される。キャッシュメモリ 3 5 4 は後述するように 1 エントリが 8 バイトで容量 2 k B の 2 ウェイセットアソシアティブキャッシュである。キャッシュメモリ 3 5 4 は、入力されたアドレスと同一アドレスの命令が過去に伸張されて保持されているか否かを調べ、保持されている場合には伸張後の V L I W 命令を出力線 3 5 9 に出力するとともに、キャッシュメモリ 3 5 4 がヒットしたことを示すためにキャッシュヒット信号線 3 5 7 上のキャッシュヒット信号をアサートする。このキャッシュヒット信号は O R 回路 3 5 5 を経て R E A D Y 信号線 2 5 6 上の R E A D Y 信号をアサートする。このとき、MUX 3 5 6 は出力線 3 5 9 上の出力を選択してキャッシュメモリ 3 5 4 から出力された伸張後の V L I W 命令をデータ線 2 5 9 上に出力する。

## 【 0 0 7 5 】

一方、キャッシュメモリ 3 5 4 がミスしたときには、命令コード伸張部 3 5 0 から出力線 3 6 0 に出力された伸張後の 8 バイトの V L I W 命令が M U X 3 5 6 によって選択され、データ線 2 5 9 に出力される。このとき、データ線 2 5 9 に出力された伸張後の命令コードは、キャッシュメモリ 3 5 4 にも転送され、対応するエントリに書込まれる。伸張器 3 5 2 での命令の伸張が終了したタイミングで、タイミング信号線 3 5 8 上のタイミング信号がアサートされる。このタイミング信号は、O R 回路 3 5 5 を経て R E A D Y 信号線 2 5 6 上の R E A D Y 信号をアサートする。

## 【 0 0 7 6 】

図 1 4 に、キャッシュメモリ 3 5 4 の詳細を示す。図 1 4 に示されるようにキャッシュメモリ 3 5 4 は 2 ウェイセットアソシアティブキャッシュであって、アドレス線 2 5 4 のうちの下位 7 ビットをインデックスとして、各ウェイに保持されているタグおよび命令コードをインデックスに基づいてサーチし、一致するタグがあるか否かを判定する。一致するタグがある場合には一致信号がそのウェイから出力される。この一致信号が O R 回路を経てキャッシュヒット信号線 3 5 7 上にキャッシュヒット信号として出力される。一方、このとき、2 つのウェイの、アドレス線 2 5 4 のインデックスにより示されるエントリからは対応の命令コード ( 6 4 ビット ) が出力され、一致信号に基づいてそのうちの一つが選択され出力バッファを経て出力線 3 5 9 上に出力される。

## 【 0 0 7 7 】

図 1 5 に、プロセッサ 1 0 が圧縮命令用のトランスレータ付メモリ 2 4 をアクセスして命令をフェッチするときのバス 4 0、R E A D Y 信号線 5 0、D C 信号 5 1 の信号変化を説明するタイミングチャートを示す。図 1 5 を参照して、キャッシュメモリ 3 5 4 がヒットしたとき、R E A D Y 信号線 5 0 上の R E A D Y 信号がバスサイクル開始後 1 クロック後にアサートされ、バスサイクル開始の 2 クロック後にバス制御回路 2 0 から D C 信号 5 1 によりバスサイクル終了がプロセッサ 1 0 に通知される。また、キャッシュメモリ 3 5 4 に保持されていた伸張後の V L I W 命令の命令コードがバスサイクル開始後 1 . 5 クロック後に圧縮命

令用のトランスレータ付メモリ 2 4 からデータ線 2 5 9 を経由してバス 4 0 に出力され、DC 信号 5 1 のサンプリングタイミングと同じバスサイクル開始後 2 クロック後にプロセッサ 1 0 に取込まれる。

#### 【 0 0 7 8 】

一方、キャッシュメモリ 3 5 4 がミスしたときには、READY 信号線 5 0 がバスサイクル開始後 1 クロック後に一度ネゲートされ、バスサイクル開始後の 3 クロック後にアサートされる。これによりバスサイクル開始の 2 および 3 クロック後はバス制御回路 2 0 からバスサイクルにウェイトサイクル 5 2 および 5 3 を挿入することが DC 信号 5 1 によりプロセッサ 1 0 に通知される。バスサイクル開始の 4 クロック後にバス制御回路 2 0 から DC 信号 5 1 によりバスサイクル終了がプロセッサ 1 0 に通知される。また、命令コード伸張部 3 5 0 により伸張された V L I W 命令の命令コードがバスサイクル開始 3 . 5 クロック後に圧縮命令用のトランスレータ付メモリ 2 4 からデータ線 2 5 9 経由でバス 4 0 に出力され、DC 信号 5 1 のサンプリングタイミングと同じくバスサイクル開始後 4 クロック後にプロセッサ 1 0 に取込まれる。

#### 【 0 0 7 9 】

図 1 6 に、1 つの 4 バイト境界にある圧縮された二つの命令 3 0 5、3 0 6 が伸張後の V L I W 命令のサブ命令のフィールドと他のフィールドとにどのように対応するかを示す。図 1 6 を参照して、圧縮された命令 3 0 5 および命令 3 0 6 との実行順序に関して、そのアドレス位置により論理的にまず命令 3 0 5 が実行され、次に命令 3 0 6 が実行される。伸張器 3 5 2 は、命令 3 0 5 と命令 3 0 6 との内容に依存して、プロセッサ 1 0 の命令 3 0 5 に対するサブ命令を L コンテナ 2 0 5 または R コンテナ 2 0 6 のフィールドに発生し、残ったコンテナに命令 3 0 6 に対応するサブ命令を発生する。つまり、2 つの命令から 1 つの命令が発生される。FM フィールド 2 0 1 a および 2 0 1 b は命令 3 0 5 と命令 3 0 6 とのオペランドに依存関係がなくかつプロセッサ 1 0 で 2 つの命令が並列実行可能なときは「0 0」となる。オペランドに依存関係があるか、またはプロセッサ 1 0 のハードウェア構成上並列実行不可能な場合、命令 3 0 5 に対応する命令が L コンテナ 2 0 5 に入るときは「1 0」、R コンテナ 2 0 6 に入るとき「0 1」の

値となる。

#### 【0080】

条件フィールド203および204はそれぞれ、命令305および命令306が条件分岐命令BRATか、条件分岐命令BRA Fか、それ以外の命令かにしたがつて「000」、「001」および「010」の値となる。

#### 【0081】

図17～図20に、圧縮された命令がサブ命令に伸張されるときに命令ビットパターンの対応例を示す。図17bに示す。「Add Ra, Rc」命令が伸張されるとサブ命令「ADD, Rx, Ry, Rz」となる。9ビットのオペコード311は異なる9ビットのオペコード321に変換される。4ビットのレジスタ番号Ra312は上位に「11」を付加してレジスタ番号Rx322およびレジスタ番号Ry323に変換される。同様に4ビットのレジスタ番号Rc314はレジスタ番号Rz324に変換される。

#### 【0082】

図18を参照して、命令「SUB Ra, #imm:4」が伸張されるとサブ命令「SUB Rx, Ry, #imm:6」になる。オペコード311とレジスタ番号Ra312との変換は図17に示す場合と同様である。4ビットのリテラル315は符号拡張され6ビットのリテラル325に変換される。

#### 【0083】

図19を参照して、命令「BRA #imm:9」が伸張されるとサブ命令「BRA #imm:18」になる。オペコード311がオペコード321に変換され、9ビットの変位316は符号拡張されて18ビットの変位326に変換される。

#### 【0084】

図20を参照して、命令「LDW Ra, @(Rb+)」が伸張されるとサブ命令「LDW Rx, @(Ry+, R0)」になる。オペコード311がオペコード321に、レジスタ番号Ra312は上位に「11」が付加されてレジスタ番号Rxに、それぞれ変換される。レジスタ番号Rb314は「ADD Ra, Rc」命令の場合と異なり上位に「00」が付加されてレジスタ番号Ryに変換される。サブ命令のレジスタ番号327はR0を示す番号「000000」となる。

## 【 0 0 8 5 】

以上のようにサブ命令では3オペランドが基本であるのに対し、圧縮された命令では2オペランドが基本である。サブ命令はオペランドとして6ビットフィールドで64個の汎用レジスタを指定できるが、圧縮された命令ではオペランドとして4ビットのフィールドで16個の汎用レジスタしか指定できない。また、変換時にはレジスタ番号の上位に「11」を付加してレジスタ番号を拡張するので、圧縮された命令でオペランドとして指定できる汎用レジスタは図21に示すように、サブ命令の汎用レジスタR48～63（153～155、…、162、163a、163b）に相当する。

## 【 0 0 8 6 】

図22を参照して、J A V A命令用のトランスレータ付メモリ25は、トランスレータ15とJ A V A命令用RAM265とを含む。

## 【 0 0 8 7 】

トランスレータ15は、J A V A命令のためのトランスレーション回路263と、アドレス線272から与えられるアドレスを右に3ビットシフトし、1/8の値としてアドレス線273に出力するためのシフト回路261と、変換機能制御信号線270上の変換機能イネーブル信号TEの値に基づいて、アドレス線273およびアドレス線272の一方を選択し、アドレス線274を介してJ A V A命令のためのトランスレーション回路263およびJ A V A命令用RAM265に与えるためのMUX262と、出力線277および出力線278の一方を選択してデータ線279に出力するためのMUX264と、R/W信号線271およびアドレス線272に接続され、J A V A命令用RAM265に対する書込があったことを検出してJ A V A命令のためのトランスレーション回路263中にあるキャッシュメモリの全エントリを無効化することを指示する制御信号を制御信号線277a上に出力するための書込検出回路266とを含む。

## 【 0 0 8 8 】

MUX262は、変換機能イネーブル信号TEが命令変換を指示している場合に、アドレス線273を選択して値が1/8にされたアドレスをアドレス線274を介してJ A V A命令用RAM265およびJ A V A命令のためのトラン

スレーション回路263に与える。またMUX262は、J A V A命令のためのトランスレーション回路263 が命令コードの変換を行なうためにJ A V A命令用RAM265を2回目以降アクセスするとき、J A V A命令のためのトランスレーション回路263からアドレス線275を介して与えられたアドレスをアドレス線274に出力する。

## 【0089】

MUX264は、J A V A命令用のトランスレータ付メモリ25から命令コードを読出す場合、J A V A命令用RAM265から出力された出力線277をそのままデータ線279に出力するか、それをJ A V A命令のためのトランスレーション回路263で変換した後の、出力線278を介して与えられる命令コードをデータ線279に出力するかを制御する。

## 【0090】

READY信号線276上のREADY信号は、J A V A命令のためのトランスレーション回路263から変換の後の命令コードを出力するタイミングを示す。READY信号は、READY信号線50に伝えられ、さらにバス制御回路20に与えられるトライステート出力信号である。

## 【0091】

書込検出回路266に与えられるR/W信号線271上のR/W信号は、J A V A命令用RAM265へデータ線279から命令コードを書込むか、読出すかを指定する信号である。J A V A命令用RAM265に対する書込があると、書込検出回路266はそれを検知して制御信号線277a上に制御信号を出力する。この制御信号がアサートされると、J A V A命令のためのトランスレーション回路263では全てのキャッシュエントリを無効化する。この動作により、J A V A命令用RAM265の古い内容に基づく古い変換結果がキャッシュから読み出されることが防止できる。

## 【0092】

プロセッサ10から見ると、J A V A命令用のトランスレータ付メモリ25のアドレスは図11に示すJ A V A命令用（直結）アドレス領域125aとJ A V A命令（トランスレータ経由）用アドレス領域125bとからなる。J A V

A命令用（直結）アドレス領域125aでJ A V A命令用のトランスレータ付メモリ25 をアクセスする場合、64kBの空間の全バイト位置に有効なメモリがある。一方、J A V A命令（トランスレータ経由）用アドレス領域125bでJ A V A命令用のトランスレータ付メモリ25をアクセスする場合、512kBの空間のアラインされた各8バイトの上位1バイトには有効なメモリがあるが、下位7バイトには有効なメモリは存在しない。

## 【0093】

プロセッサ10 がたとえばアドレスH' 20050300から圧縮されたV L I W命令をフェッチして実行するとき、J A V A命令用のトランスレータ付メモリ25からはJ A V A命令が1または複数の8バイトのV L I W命令に伸張されて出力され、プロセッサ10はJ A V A命令1バイトにつきPC値を8バイト進めて1または複数のV L I W命令を実行する。なお、J A V A命令についてはTim Lindholm and Frank Yellin, "The Java Virtual Machine Specification Second Edition," Sun Microsystems, Inc., 1999に詳しく記載されている。

## 【0094】

図23を参照して、J A V A命令のためのトランスレーション回路263は、命令コード変換部370と、キャッシュメモリ374と、アドレス線274から入力されたアドレスを命令コード変換部370およびキャッシュメモリ374に与え、さらにアドレス線275上に出力するためのアドレス制御回路373と、キャッシュメモリ374の出力線379および命令コード変換部370の出力線380を受け、いずれか一方から与えられる命令コードを選択してデータ線279に出力するためのMUX376と、キャッシュメモリ374からキャッシュヒット信号線377を介して与えられるヒット信号と、命令コード変換部370から信号線378を介して与えられる伸張完了を示すタイミング信号とを受け、出力がREADY信号線276に接続されたOR回路375とを含む。命令コード変換部370は、キュー371と変換器372とを含む。

## 【0095】

図24を参照して、キャッシュメモリ374は1エントリが8バイトで容量が2kBのダイレクトマップキャッシュである。キャッシュメモリ354の各エン

トリは、Vビット387およびUビット382との2つのバリッドビットと、比較対照アドレスのタグ383と、命令コードであるデータ384とを含む。

【0096】

図22を参照して、アドレス線272から与えられたH' 205000-H' 200Cffffの一つを示す19ビットのアドレスは、シフト回路261によって3ビット右にシフトされて1/8にアドレス変換され、その有効16ビットがアドレス線274およびアドレス制御回路373を介してJ A V A命令のためのトランスレーション回路263および命令コード変換部370に与えられる。

【0097】

図24を参照して、キャッシュメモリ374では、このアドレスは上位8ビットのタグ385と下位8ビットのインデックス386とに分解される。インデックス386はエントリの選択に使用される。タグ385は各エントリから読み出されたタグ383との比較対照に用いられる。Vビット387が有効を示し、タグ383の値とタグ385の値とが等しいエントリがあれば、キャッシュヒットとなり、ヒット信号がアサートされる。一方当該エントリのデータ384が命令コードとして出力線379に出力される。図23に示されるキャッシュヒット信号線377上に出力されるキャッシュヒット信号は、OR回路375を経てR E A D Y信号線276上のR E A D Y信号をアサートする。

【0098】

再び図23を参照して、命令コード変換部370では、キュー371が一旦J A V A命令をキューに入れた後、変換器372が1バイトずつJ A V A命令を読み出し、V L I W命令に変換して命令コード変換部370の出力線380上に出力する。変換器372の処理クロック数はJ A V A命令の長さおよび種類、並びに変換後のV L I W命令の数および種類に依存して変わる。

【0099】

キャッシュヒット信号線377上のキャッシュヒット信号がアサートされた場合、M U X 3 7 6は出力線379を選択して、キャッシュメモリ374から出力された、変換後のV L I W命令をデータ線279上に出力する。一方、キャッシュヒット信号がアサートされなかったとき、すなわちキャッシュメモリ374が

ミスしたときには、MUX 376は命令コード変換部370が命令コード変換部370の出力線380上に出力した変換後の8バイトのVLIW命令を選択して、データ線279上に出力する。このとき、命令コード変換部370の出力線380上のVLIW命令はキャッシュメモリ374にも転送され、対応するエントリに書込まれる。また、変換が完了し命令コード変換部370の出力線380上にVLIW命令が出力されるタイミングで信号線378上のタイミング信号がアサートされ、このタイミング信号はOR回路375を経てREADY信号線276上のREADY信号をアサートする。

#### 【0100】

なお、JAVA命令を複数のVLIW命令に変換する場合、命令コード変換部370はプロセッサ10からのアクセスに関係なく変換後のVLIW命令を全てキャッシュメモリ374に登録する。

#### 【0101】

JAVA命令は、バイト単位で命令長が可変であり、メモリ上のどのアドレスにも配置可能である。そのため2バイト以上のJAVA命令をメモリから読出す場合、JAVA命令のアドレス位置に依存してJAVA命令用RAM265を2回以上アクセスしなければならない可能性があることに注意が必要である。

#### 【0102】

たとえば、2バイトのJAVA命令J2が8バイト境界の最後のアドレスである( $8n+7$ )番地にある場合を考える。このとき、JAVA命令用RAM265では、1回目のアクセスで( $8n+7$ )番地からJ2の最後の1バイトを読出し、2回目のアクセスで $8(n+1)$ 番地からJ2の次の1バイトを読出す必要がある。この2回目のアクセスのためのアドレス $8(n+1)$ 番地は、JAVA命令のためのトランスレーション回路263からアドレス線275に出力される。

#### 【0103】

既に述べたように、変換器372によって変換されたVLIW命令は、命令コード変換部370の出力線380を経てキャッシュメモリ374に1つずつ登録される。1つのJAVA命令を複数のVLIW命令に変換してキャッシュメモリ

374に登録する場合には、これら複数のVLIW命令が常に組みになってキャッシュメモリ374に保持されているように配慮をする必要がある。そのために、先頭のVLIW命令以外のエントリでは、Uビット382の「1」とする。こうすることにより、これらを含むエントリが、キャッシュメモリ374においてソフトウェアの関与なしに無効化されたり、キャッシュメモリ374から追い出されたりすることを禁止する。また、1つのJAVA命令を複数のVLIW命令に変換してキャッシュメモリ374に登録する場合、キャッシュが一杯で先頭のVLIW命令以外のVLIW命令がキャッシュメモリに登録できない場合もあり得る。この場合には、FULL信号381をアサートし、このアサート信号にตอบสนองしてソフトウェア制御によって先頭のVLIW命令を含むエントリを無効化する。

#### 【0104】

これは、1つのJAVA命令を複数のVLIW命令に変換した場合、たとえばVLIW命令AとVLIW命令Bとの間で割込が受けられ、割込からの復帰後にVLIW命令Bが実行されるときに、VLIW命令Bがキャッシュメモリにないという事態となって、1つのJAVA命令の途中から命令変換が行なわれることになり間違った変換が行なわれてしまうことを防ぐためである。JAVA命令を含め、プロセッサの命令では一般に、途中のバイトに即値データがあり、命令の先頭バイトから見ない限り命令を正しく認識できないためである。なお、FULL信号381がアサートされたときに、ソフトウェア制御によってUビット382が「1」であるが対応する先頭のVLIW命令がキャッシュにないために安全に無効化が可能なエントリがある場合、それらのエントリを無効化することにより、キャッシュメモリ374の空きエントリを増加させ、そこでFULL信号381をアサートするきっかけとなったVLIW命令をキャッシュメモリ374に登録するようにしてもよい。

#### 【0105】

さて、キャッシュメモリ374において従来の通常のキャッシュメモリと同様の方法でエントリの無効化または新規登録のためのエントリの追出しを行なうと、割込がなくても変換器372で1つのJAVA命令の途中から命令変換が行

なわれる可能性がある。これを防ぐために、本実施の形態の装置ではキャッシュメモリ 374 の各エントリに U ビット 382 を設けた。1 つの J A V A 命令から変換された、連続するアドレスにある複数の V L I W 命令がキャッシュメモリ 374 に登録されたとき、先頭の V L I W 命令以外の V L I W 命令に対応するエントリでは U ビット 382 が「1」とされる。U ビット 382 が「1」のエントリは、ソフトウェアによる操作なしに無効化されたり、キャッシュメモリ 374 から追出されることはない。先にも述べたように U ビット 382 が「1」のエントリの無効化または追出しは、そのエントリに含まれる V L I W 命令とともに J A V A 命令から変換された複数の V L I W 命令の先頭の V L I W 命令がキャッシュメモリ 374 内に存在しないときのみ可能である。これはソフトウェアにより管理される。

#### 【0106】

これにより、1 つの J A V A 命令から変換された複数の V L I W 命令の先頭の V L I W 命令がキャッシュメモリ 374 に登録されたままで、後続の V L I W 命令がキャッシュメモリ 374 に登録されてない状態が発生し、変換器 372 が J A V A 命令の途中から変換を開始することを防ぐ。

#### 【0107】

図 25 を参照して、命令コード変換部 370 が変換する J A V A 命令と V L I W 命令との関係は以下のとおりである。たとえば 1 バイトの J A V A 命令 401 は 8 バイトの V L I W 命令 411 に変換される。J A V A 命令 401 は 1 バイトなので、この J A V A 命令 401 は J A V A 命令用 R A M 265 を 1 回アクセスするだけで読出すことができる。変換後の V L I W 命令 411 も 8 バイト境界に位置するため、キャッシュメモリ 374 の 1 エントリに登録される。

#### 【0108】

2 バイトの J A V A 命令 402 は 2 つの 8 バイトの V L I W 命令 412 a、412 b に変換される。このとき、V L I W 命令 412 b を含むエントリでは U ビット 382 が「1」となる。

#### 【0109】

5 バイトの J A V A 命令 404 は 3 つの 8 バイトの V L I W 命令 414 a、4

14bおよび414cに変換される。VLIW命令414bおよび414cを含むエントリでは、Uビット382が共に「1」となる。

【0110】

JAVA命令403、JAVA命令406～408などについても同様で、JAVA命令403はVLIW命令413に、JAVA命令406はVLIW命令416に、JAVA命令407はVLIW命令417に、JAVA命令408は3つの8バイトのVLIW命令418a～418cに、それぞれ変換される。VLIW命令418bおよび418cを含むエントリでは、Uビット382が共に「1」となる。

【0111】

図26～図31は、変換器372でJAVA命令をVLIW命令に変換したときの具体的な例を示す。図26を参照して、1バイトのJAVA命令「iadd」はサブ命令「LDW R61, @(R63+, R0)」とサブ命令「ADD R62, R62, R61」とを順に実行する1つのVLIW命令に変換される。「iadd」はスタックトップから1番目のデータと2番目のデータとである2つの32ビット整数を加算してスタックに書き戻すJAVA命令である。プロセッサ10ではスタックトップのデータはレジスタR62に配置され、レジスタR63がスタックトップの次のデータのアドレスを示す。したがって、プロセッサ10では、JAVA命令「iadd」のオペレーションを、スタックトップから2番目の32ビットデータをレジスタR61にロードしてレジスタR62を4インクリメントするオペレーションを行なうサブ命令「LDW R61, @(R63+, R0)」と、レジスタR62とレジスタR61との2つの32ビット整数を加算してその結果をレジスタR62に書込むオペレーションを行なうサブ命令「ADD R62, R62, R61」とでエミュレートすることができる。このとき、PC値については、1つのVLIW命令を実行したことによりプロセッサ10のPC値を8番地進めることで、JAVA命令「iadd」の実行に対応してPC値を1進めることをエミュレートする。

【0112】

図27を参照して、2バイトのJAVA命令「iload」は、1つのサブ命令「ADD/CN R50, #(0||vindex)」を持つVLIW命令と、サブ命令「STW R62, @(R63-

、R4)」とサブ命令「LDW R62, @(R10, R50)」とを順次に行行するV L I W命令に交換される。「iload」はローカル変数領域から32ビット整数をフェッチしてスタックトップに保存するJ A V A命令である。プロセッサ10ではこれを、レジスタR50にローカル変数のインデックス値をロードするサブ命令「ADD/CN R50, #(0||vindex)」と、レジスタR62をスタックトップから2番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、スタックトップであるレジスタR62にローカル変数領域からデータをロードするサブ命令「LDW R62, @(R10, R50)」とに分解することによりエミュレートする。

#### 【0113】

なお、ここでレジスタR4には、値「-4」を保持し、レジスタR10にはローカル変数領域のベースアドレスを保持する。P C値については2つのV L I W命令を実行したことによりプロセッサ10のP C値を16番地進めることで、J A V A命令「iload」の実行に対応してP C値を2番地進めることをエミュレートする。

#### 【0114】

図28を参照して、3バイトのJ A V A命令「ifeq」は、サブ命令「ADD R62, R61, R0」とサブ命令「NOP」とを並列に行行するV L I W命令と、サブ命令「LDW R62, @(R63+, R0)」とサブ命令「NOP」とを並列に行行するV L I W命令と、1つのサブ命令「BRATZR/CN R62, #(s||branchbyte1||branchbyte2)」を実行するV L I W命令とに分解される。命令「ifeq」はスタックトップのデータが「0」なら分岐するJ A V A命令である。プロセッサ10ではこれを、レジスタR62をレジスタR61にコピーするサブ命令「ADD R62, R61, R0」と、スタックトップから2番目のデータをレジスタR62にポップするサブ命令「LDW R62, @(R63+, R0)」と、レジスタR61がゼロなら分岐するサブ命令「BRATZR/CN R62, #(s||branchbyte1||branchbyte2)」とに分解し、これら3つのサブ命令と2つのNOP命令とを組合せて、計3つのV L I W命令でエミュレートする。P C値については、3つのV L I W命令を実行したことによりプロセッサ10のP C値を24番地進めることで、J A V A命令「ifeq」の実行に対応してP C値を3番地進めることをエミュレートする。

## 【0115】

図29を参照して、5バイトのJ A V A命令「jsr\_w」は、1つのサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」からなるV L I W命令と、サブ命令「STW R62, @(R63-, R4)」とサブ命令「BSR R10」とを順次に実行するV L I W命令と、サブ命令「BRA #3」とサブ命令「NOP」とを並列に実行するV L I W命令とに変換される。「jsr\_w」は戻り先アドレスをスタックにプッシュして4バイトで指定されたアドレスのサブルーチンヘジャンプするJ A V A命令である。プロセッサ10ではこの命令を、ジャンプ先アドレスをレジスタR10にロードするサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」と、スタックトップのレジスタR62の値をスタックトップから2番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタR62に保存してレジスタR10で指定されたアドレスのサブルーチンヘジャンプするサブ命令「JSR R10」と、サブルーチンから復帰した後に2つのV L I W命令をスキップするための分岐を行なうサブ命令「BRA #3」と、サブ命令「NOP」とに分解してエミュレートする。P C値については、3つのV L I W命令を実行し、2つのV L I W命令をスキップする分岐を行なうことによりプロセッサ10のP C値を40番地進めることで、J A V A命令「jsr\_w」の実行に対応してP C値を5番地進めることをエミュレートする。

## 【0116】

図30と図31とは、複雑なJ A V A命令を、V L I W命令からなりかつJ A V A命令の機能を実行するサブルーチンをコールするV L I W命令に変換する例を示す。図30を参照して、この例では、浮動小数点数の加算を行なうJ A V A命令「fadd」を、スタックトップのレジスタR62の値をスタックトップから2番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタR62に保存して#faddで指定されたアドレスのサブルーチンにジャンプするサブ命令「JSR #fadd」とを順次に実行するV L I W命令に変換する。プロセッサ10では、「fadd」のオペレーションをサブルーチン中に行ない、P C値の更新に関しては、一つのV L I W命令を実行することで

プロセッサ10のPC値を8番地進めて「fadd」に対応するPC値を1番地進めることをエミュレートする。

#### 【0117】

図31に示す例では、テーブルジャンプを行なうJ A V A命令「tableswitch」を、スタックトップのレジスタR62の値をスタックトップから2番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタR62に保存して#tableswitchで指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR #tableswitch」とを順次に実行するV L I W命令に変換する。プロセッサ10では、「tableswitch」のオペレーションとPC値の更新との両方をサブルーチン中でJ A V A命令「tableswitch」で指定された各種パラメータをアクセスしてエミュレートする。このとき、プロセッサ10は、J A V A命令用（直結）アドレス領域125aでJ A V A命令用のトランスレータ付メモリ25をアクセスしてJ A V A命令「tableswitch」で指定された各種パラメータをデータとして読出す。

#### 【0118】

図32を参照して、プロセッサ10がJ A V A命令用のトランスレータ付メモリ25を使用してJ A V A命令で記述されたプログラムをエミュレートする時の、プロセッサ10で実行される処理のアルゴリズムは、以下のような制御構造を有する。まずプロセッサ10は、ROM23からJ A V AプログラムをJ A V A命令用のトランスレータ付メモリ25にロードする（420）。次にプロセッサ10は、J A V Aの実行環境を作り、エミュレーションのための初期設定を行なう（421）。この後、プロセッサ10はJ A V A命令用のトランスレータ付メモリ25のアドレスにジャンプすることによりエミュレーションを開始する（422）。

#### 【0119】

キャッシュメモリ374をアクセスして、ヒットならば制御はステップ428に進み、ミスならば制御はステップ424に進む（423）。キャッシュミスの場合、J A V A命令用RAM265からJ A V A命令をフェッチし、命令コード変換部370でJ A V A命令をネイティブのV L I W命令に変換する（424

）。キャッシュメモリ374に変換後のネイティブ命令を登録可能ならばステップ427に進む。キャッシュメモリ374の対応するエントリがフルで登録不可の場合にはFULL信号381をアサートしてプロセッサ10に対して割込を要求し、ソフトウェア制御でキャッシュメモリ374に変換後のネイティブ命令を登録するための空きエントリを作成する(425、426)。

#### 【0120】

次に、プロセッサ10で変換後のネイティブ命令を実行してJAVA命令をエミュレートする(428)。エミュレーションが終了でないならばPC値を進めてキャッシュメモリ374をアクセスする。エミュレーションが終了であればJAVA命令用のトランスレータ付メモリ25の動作を終了してネイティブ命令のプログラムに戻る。

#### 【0121】

ここで、ステップ420～422、426の処理は、JAVA命令を変換したネイティブ命令での処理ではなく、ネイティブ命令用RAM21に存在するネイティブのVLIW命令のプログラムを実行することにより行なう。

#### 【0122】

非ネイティブ命令X用のトランスレータ付メモリ26は、8バイトの固定長の命令Xをプロセッサ10のVLIW命令に変換して実行するためのものである。プロセッサ10から非ネイティブ命令X用のトランスレータ付メモリ26を見た場合、そのアドレスは図11に示す非ネイティブ命令X(直結)用アドレス領域126aと非ネイティブ命令X(トランスレータ経由)用アドレス領域126bとからなる。どちらのアドレス領域で非ネイティブ命令X用のトランスレータ付メモリ26をアクセスしても64kBの空間の全バイト位置に有効なメモリが存在する。アドレス領域126bで非ネイティブ命令X用のトランスレータ付メモリ26をアクセスする場合、命令Xがトランスレータ16でVLIW命令に変換されて出力される。トランスレータ16の詳細は図12に示されるトランスレータ14とほぼ同様である。ただし、トランスレータ16においては、図12に示されるアドレス変換器241、MUX242およびアドレス制御回路373に対応する回路を持たず、アドレスが変換されずに入力されること、キャッシュメ

モリにUビット382とFULL信号381とに対応する機能がないこと、および常に1つの命令Xが1つのVLIW命令に変換されることがトランスレータ14と異なる。

#### 【0123】

以上の実施の形態の装置では、キャッシュメモリの1エントリに1つのVLIW命令を保持する場合について説明した。しかしこの発明はそうした構成には限定されず、1エントリに複数のVLIW命令を保持するようにしてもよい。また、上記実施の形態の装置ではJAVA命令の1バイトを1つのVLIW命令に対応させてエミュレートしているが、JAVA命令の1バイトについて2または3のVLIW命令命令を対応させてエミュレートするようにしてもよい。

#### 【0124】

さらに、上記実施の形態では変換後の命令をキャッシュに保持する例を述べた。しかしこの発明はそうした構成に限定されるわけではなく、このキャッシュメモリはなくてもよい。また、キャッシュではなく変換後の複数のVLIW命令を一時的に保持し、使用後は毎回保持内容を無効化するバッファメモリを用いてもよい。

#### 【0125】

今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

#### 【0126】

##### 【発明の効果】

以上のように請求項1に記載の発明によれば、プロセッサ本体の構成を変更することなく非ネイティブ命令をネイティブ命令に変換してプロセッサで実行することができ、かつ高速に変換後の命令を出力できるので、効率的に非ネイティブ命令で記述されたプログラムのエミュレートを行なうことができる。

#### 【0127】

請求項2に記載の発明によれば、請求項1に記載の発明の効果に加えて、非ネ

イティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がない。そのため非ネイティブ命令で記述されたプログラムのエミュレートが容易に行える。

## 【 0 1 2 8 】

請求項 3 に記載の発明によれば、請求項 2 に記載の発明の効果に加えて、非ネイティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がなくエミュレートがさらに容易に行え、かつプログラムを保存するためのメモリが小さくてすみ、ハードウェアのコストの増大が回避できる。

## 【 0 1 2 9 】

請求項 4 に記載の発明によれば、請求項 3 に記載の発明の効果に加えて、非ネイティブ命令である第 2 の命令体系の命令のエミュレートを容易に行なうことができる。さらに、非ネイティブ命令の長さも第 1 の命令体系のネイティブ命令の長さでエミュレートでき、非ネイティブ命令で記述されたプログラムのエミュレートがさらに容易になる。

## 【 0 1 3 0 】

請求項 5 に記載の発明によれば、請求項 1 に記載の発明の効果に加えて、命令の変換効率がよく、効率的に非ネイティブ命令で記述されたプログラムのエミュレートを行なうことができる。

## 【 0 1 3 1 】

請求項 6 に記載の発明によれば、請求項 1 に記載の発明の効果に加えて、命令の変換が容易であり、かつ、非ネイティブ命令のプログラムカウンタ値を変換後のネイティブ命令の長さでエミュレートすることができるので、容易に、かつ効率的に、非ネイティブ命令で記述されたプログラムのエミュレートを行なうことができる。

## 【 0 1 3 2 】

請求項 7 に記載の発明によれば、請求項 6 に記載の発明の効果に加えて、非ネイティブ命令とネイティブ命令のサブ命令とを対照させることにより、変換が容易に行える。そのため、非ネイティブ命令で記述されたプログラムのエミュレートをさらに容易に行なうことができる。

## 【 0 1 3 3 】

請求項 8 に記載の発明によれば、変換後のネイティブ命令を高速にプロセッサに与えることができる。さらに、複数のネイティブ命令の同時無効化など、無効化のための条件が複雑な場合でも容易に対処できる。その結果、非ネイティブ命令で記述されたプログラムのエミュレートが高速かつ確実に行える。

## 【 0 1 3 4 】

請求項 9 に記載の発明によれば、請求項 8 に記載の発明の効果に加えて、ソフトウェアの責任において安全に保持手段の保持内容を維持することができるので、非ネイティブ命令で記述されたプログラムのエミュレートをより確実に行える。

## 【 0 1 3 5 】

請求項 1 0 に記載の発明によれば、請求項 8 に記載の発明の効果に加えて、無効化しても安全なエントリを無効化することをソフトウェア処理によって明示的に指示して、新たな命令を保持可能とすることができるので、保持手段の保持内容を安全かつ確実に維持することができる。その結果、非ネイティブ命令で記述されたプログラムのエミュレートをより確実にかつ高速に行える。

## 【 0 1 3 6 】

請求項 1 1 に記載の発明によれば、プロセッサの変更なしに、非ネイティブ命令により記述されたプログラムをプロセッサのネイティブ命令で実行できる。

## 【 0 1 3 7 】

請求項 1 2 に記載の発明によれば、請求項 1 1 に記載の発明の効果に加えて、第 2 の命令体系で記述されたプログラムを他のメモリに転送したり、その内容を解析したりすることが可能になり、プログラム開発および保守が容易に行える。

## 【 0 1 3 8 】

請求項 1 3 に記載の発明によれば、請求項 1 1 に記載の発明の効果に加えて、命令変換時と、それ以外のときとで、トランスレータ付命令メモリに関して異なるメモリマップを使用することができ、トランスレータ付命令メモリの用途に応じて適切にアクセスすることが可能になる。

## 【 0 1 3 9 】

請求項 1 4 に記載の発明によれば、請求項 1 1 に記載の発明の効果に加えて、高速に変換後の命令を出力することができるので、非ネイティブ命令により記述されたプログラムを高速にエミュレートすることができる。

【 0 1 4 0 】

請求項 1 5 に記載の発明によれば、プロセッサの構成を変更することなく、第 2 の命令体系の命令で記述されたプログラムをこのプロセッサで実行することが可能となり、ハードウェアの増大を招くことなく異なる命令体系の命令で記述されたプログラムをエミュレートできる。

【 0 1 4 1 】

請求項 1 6 に記載の発明によれば、請求項 1 5 に記載の発明の効果に加えて、プロセッサ本体の変更なしに非ネイティブ命令もネイティブ命令も区別なくプロセッサでデコードし実行することができ、非ネイティブ命令で記述されたプログラムのエミュレートがより容易になる。

【 0 1 4 2 】

請求項 1 7 に記載の発明によれば、請求項 1 6 に記載の発明の効果に加えて、プロセッサでは命令が非ネイティブ命令の場合もネイティブ命令の場合も区別なくフェッチしてデコードすることができ、非ネイティブ命令で記述されたプログラムのエミュレートがより容易になる。

【 0 1 4 3 】

請求項 1 8 に記載の発明によれば、請求項 1 6 に記載の発明の効果に加えて、複数種類の非ネイティブ命令で記述されたプログラムを、区別なくネイティブ命令を用いて実行でき、より幅広い範囲の非ネイティブ命令で記述されたプログラムのエミュレートを容易に行なうことができる。

【図面の簡単な説明】

【図 1】 従来の旧命令エミュレーション機能付データ処理装置の概略ブロック図である。

【図 2】 本発明の 1 実施の形態にかかる命令トランスレータ機能付メモリを備えたデータ処理装置のブロック図である。

【図 3】 図 2 に示すプロセッサ 1 0 のブロック図である。

【図4】 プロセッサ10が有するレジスタの一覧を表形式で示す図である。

【図5】 プロセッサ10の制御レジスタ170の詳細を示す図である。

【図6】 プロセッサ10のパイプライン処理機構を説明するための図である。

【図7】 プロセッサ10で実行可能なVLIW命令のフォーマットを示す図である。

【図8】 プロセッサ10で実行可能なVLIW命令のサブ命令のフォーマットを示す図である。

【図9】 プロセッサ10のサブ命令のパイプライン処理方法を示すための図である。

【図10】 プロセッサ10で2つのサブ命令をパイプライン処理する方法を説明するための図である。

【図11】 図2に示すデータ処理装置のメモリマップを示す図である。

【図12】 図2のデータ処理装置内の、圧縮命令を伸張する圧縮命令用のトランスレータ付メモリ24の詳細を示す図である。

【図13】 圧縮命令を変換するためのトランスレーション回路243の詳細を示す図である。

【図14】 図13に示すキャッシュメモリ354の詳細を示す図である。

【図15】 図2に示すデータ処理装置のバスサイクルのタイミングチャートである。

【図16】 2つの圧縮命令をVLIW命令に伸張するときのビットフィールドの対応を示す図である。

【図17】 圧縮命令を伸張する具体例を示す図である。

【図18】 圧縮命令を伸張する具体例を示す図である。

【図19】 圧縮命令を伸張する具体例を示す図である。

【図20】 圧縮命令を伸張する具体例を示す図である。

【図21】 本実施の形態のデータ処理装置において、圧縮命令でアクセスできる汎用レジスタの一連を示す図である。

【図22】 図2に示した、J A V A命令を変換するJ A V A命令用のトランスレータ付メモリ25の詳細を示す図である。

【図23】 J A V A命令のためのトランスレーション回路263の詳細を示す図である。

【図24】 ソフトウェアでしか無効化できないエントリを備えたキャッシュメモリ374の詳細を示す図である。

【図25】 J A V A命令を1または複数のV L I W命令に変換するときの命令間の対応を示す図である。

【図26】 J A V A命令をV L I W命令に変換する具体例を示す図である。

【図27】 J A V A命令をV L I W命令に変換する具体例を示す図である。

【図28】 J A V A命令をV L I W命令に変換する具体例を示す図である。

【図29】 J A V A命令をV L I W命令に変換する具体例を示す図である。

【図30】 J A V A命令をV L I W命令に変換する具体例を示す図である。

【図31】 J A V A命令をV L I W命令に変換する具体例を示す図である。

【図32】 図2に示すデータ処理装置がJ A V A命令のエミュレーションを行なう際の処理の流れを示すフローチャートである。

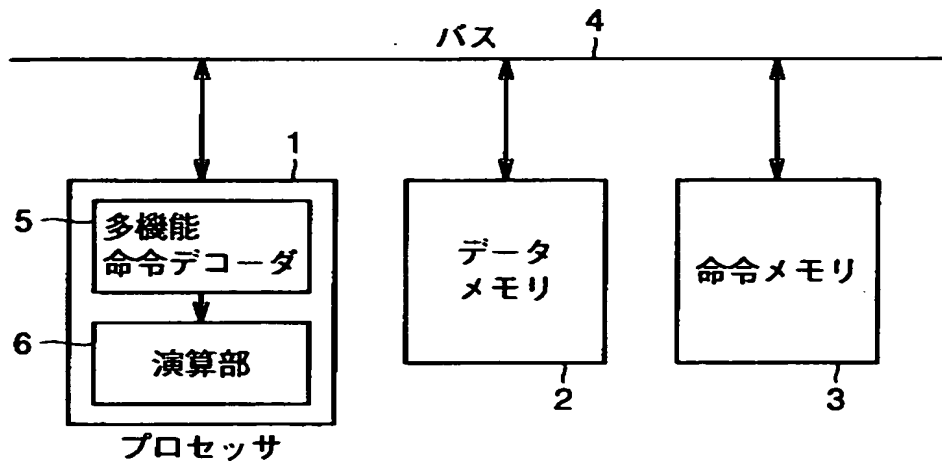
#### 【符号の説明】

10 プロセッサ、 20 バス制御回路、 40 バス、 14、15、16 命令トランスレータ、 24 圧縮命令用のトランスレータ付メモリ、 21 ネイティブ命令用RAM、 22 データ用RAM、 23 ROM、 25 J A V A命令用のトランスレータ付メモリ、 26 非ネイティブ命令X用トランスレータ付メモリ、 243、263 トランスレーション回路、 245 圧縮命令用RAM、 265 J A V A命令用RAM、 354、374 キャッシュメモリ、 35

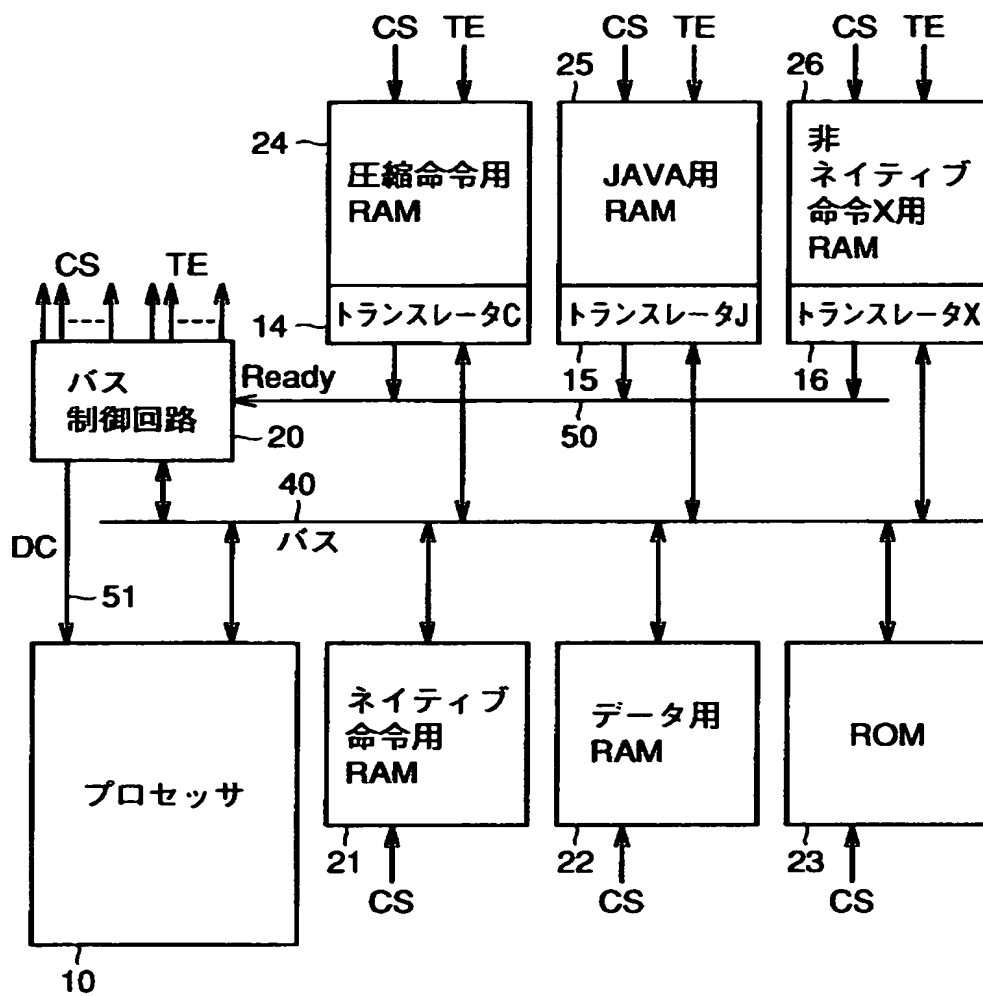
0、3 7 0 命令コード伸張部。

【書類名】 図面

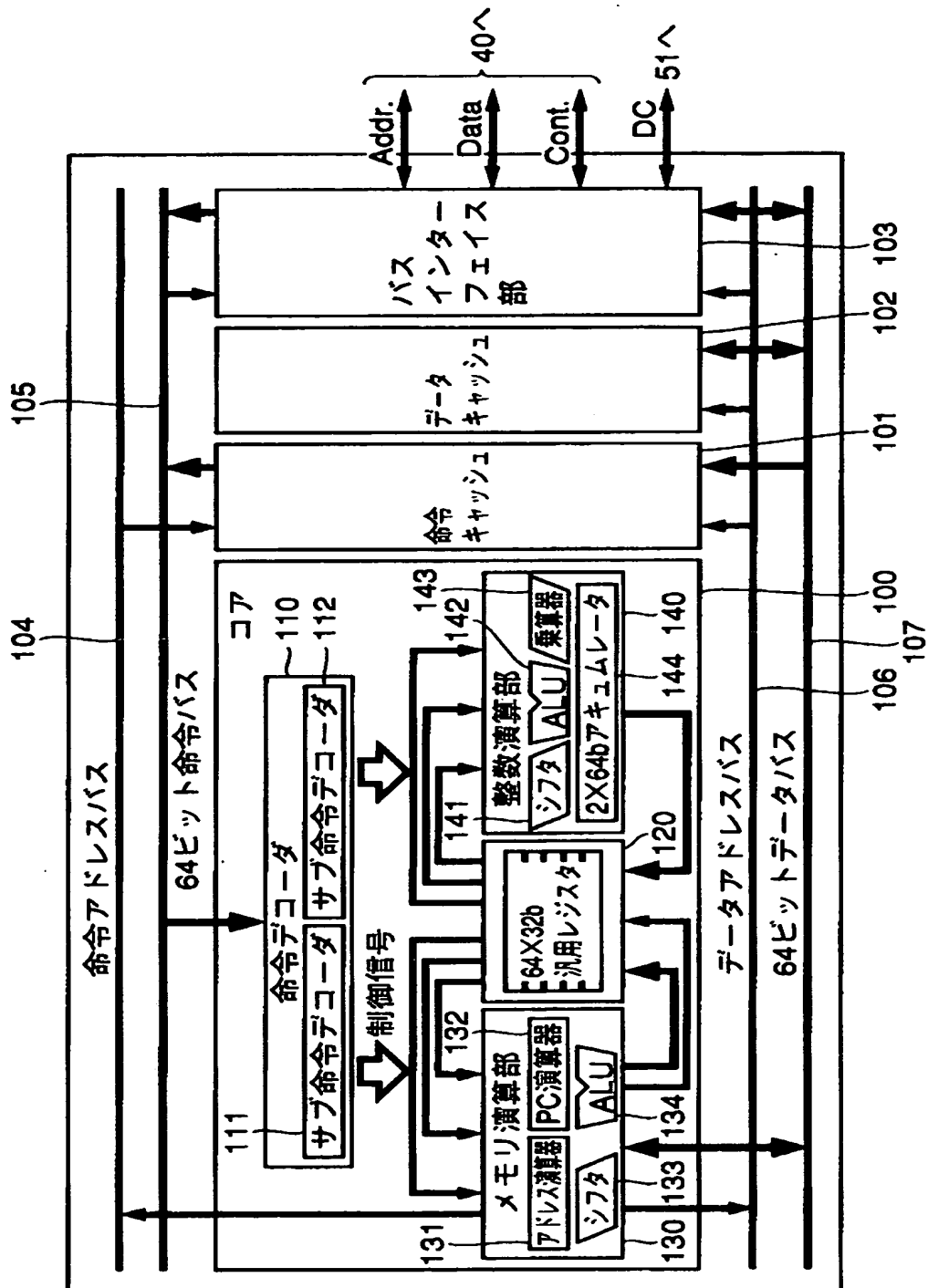
【図1】



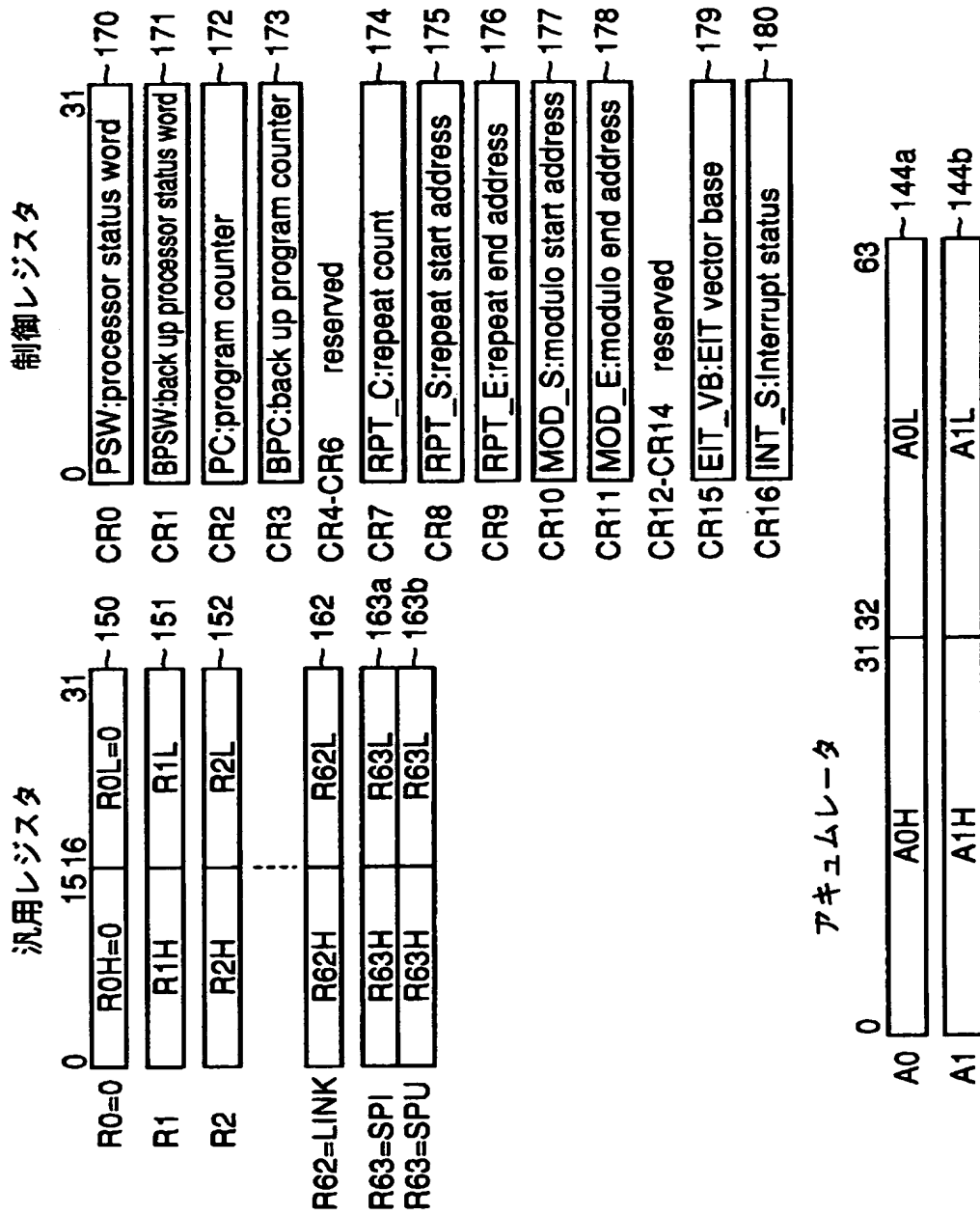
【図2】



【図 3】



【図 4】



【図 5】

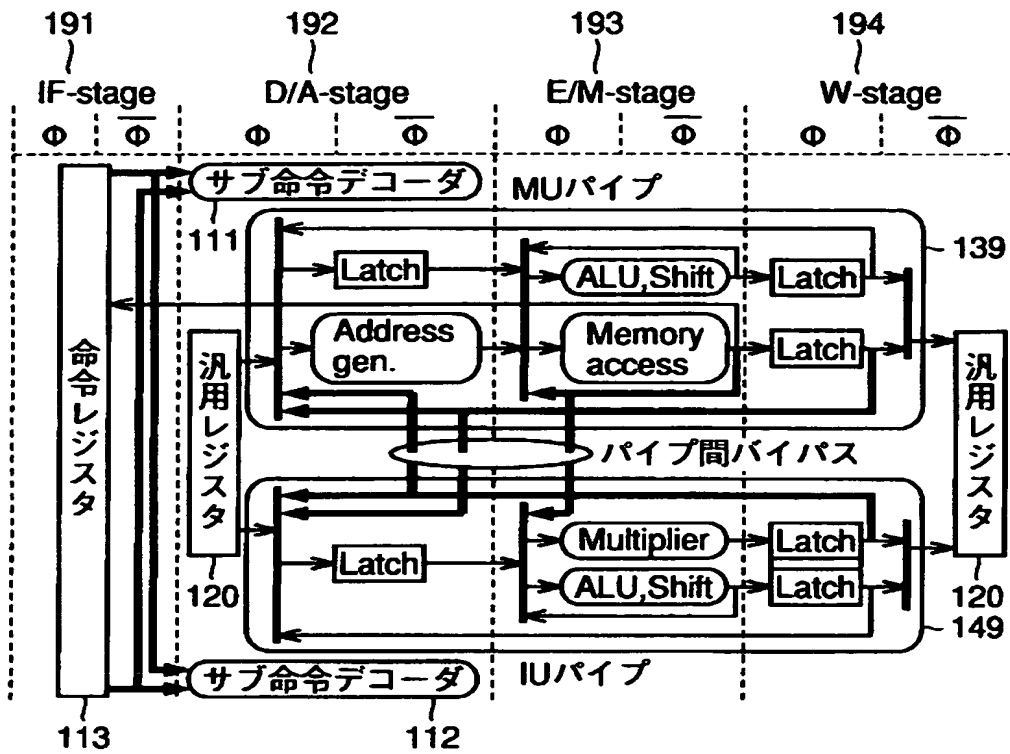
|        |   |   |   |      |   |   |   |     |    |    |   |    |   |   |   |
|--------|---|---|---|------|---|---|---|-----|----|----|---|----|---|---|---|
| 0 170a |   |   |   | 170b |   |   |   | 7 8 |    |    |   | 15 |   |   |   |
| SM     | 0 | 0 | 0 | 0    | 0 | 0 | 0 | IE  | RP | MD | 0 | 0  | 0 | 0 | 0 |

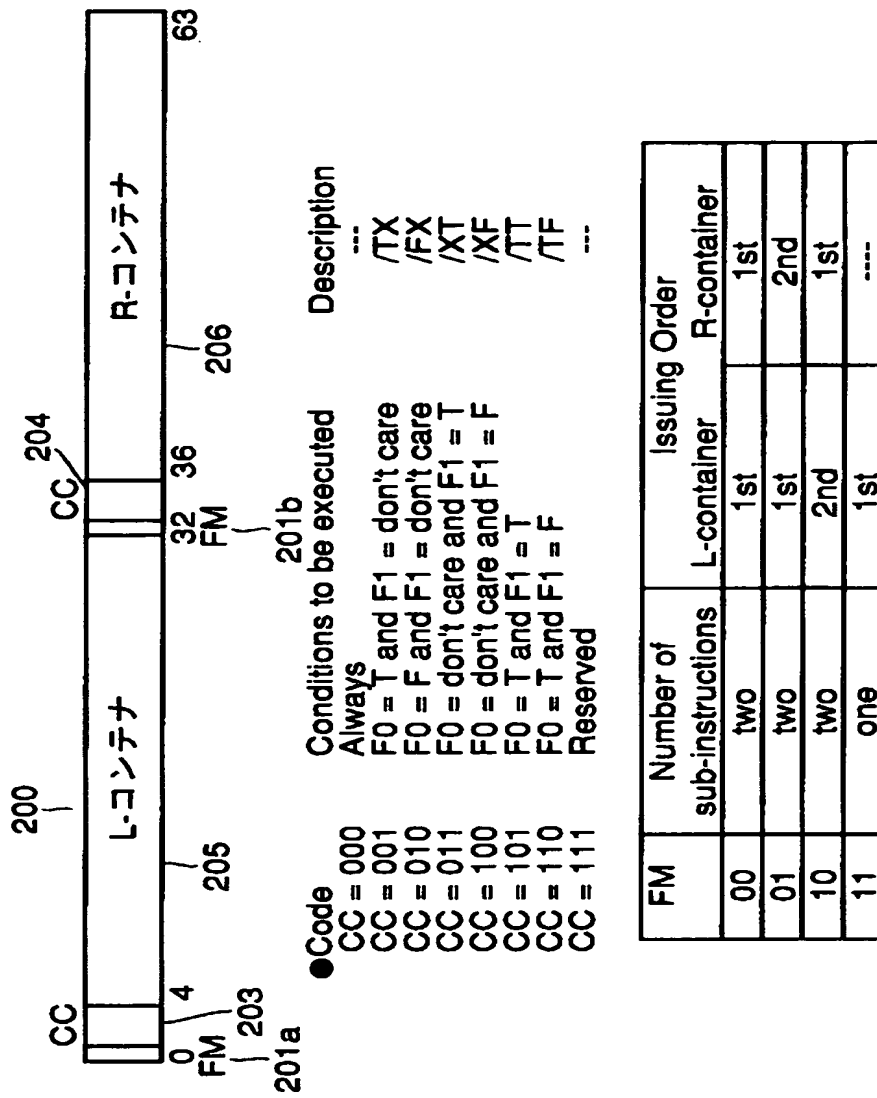
|         |    |   |   |      |   |   |    |       |    |   |    |    |    |   |    |
|---------|----|---|---|------|---|---|----|-------|----|---|----|----|----|---|----|
| 16 170c |    |   |   | 170d |   |   |    | 23 24 |    |   |    | 31 |    |   |    |
| 0       | F0 | 0 | 0 | F1   | 0 | 0 | F2 | 0     | F3 | 0 | F4 | 0  | F5 | 0 | F7 |

- SM: Stack mode  
SM=0 Interrupt mode.SPI is used.  
SM=1 User mode.SPU is used.
- IE: Interrupt enable  
IE=0 Interrupts are masked.  
IE=1 Interrupts are accepted.
- RP: Repeat enable  
RP=0 A block repeat is inactive.  
RP=1 A block repeat is active.
- MD: Modulo enable  
MD=0 Modulo addressing is disabled.  
MD=1 Modulo addressing is enabled.
- F0 : Execution control flag #0
- F1 : Execution control flag #1
- F2 : General flag
- F3 : General flag
- F4(S) : Saturation flag
- F5(V) : Overflow flag
- F6(VA) : Accumulated Overflow flag  
VA is cleared by a reset interrupt and MVTSYS instruction.
- F7(C) : Carry/Borrow flag

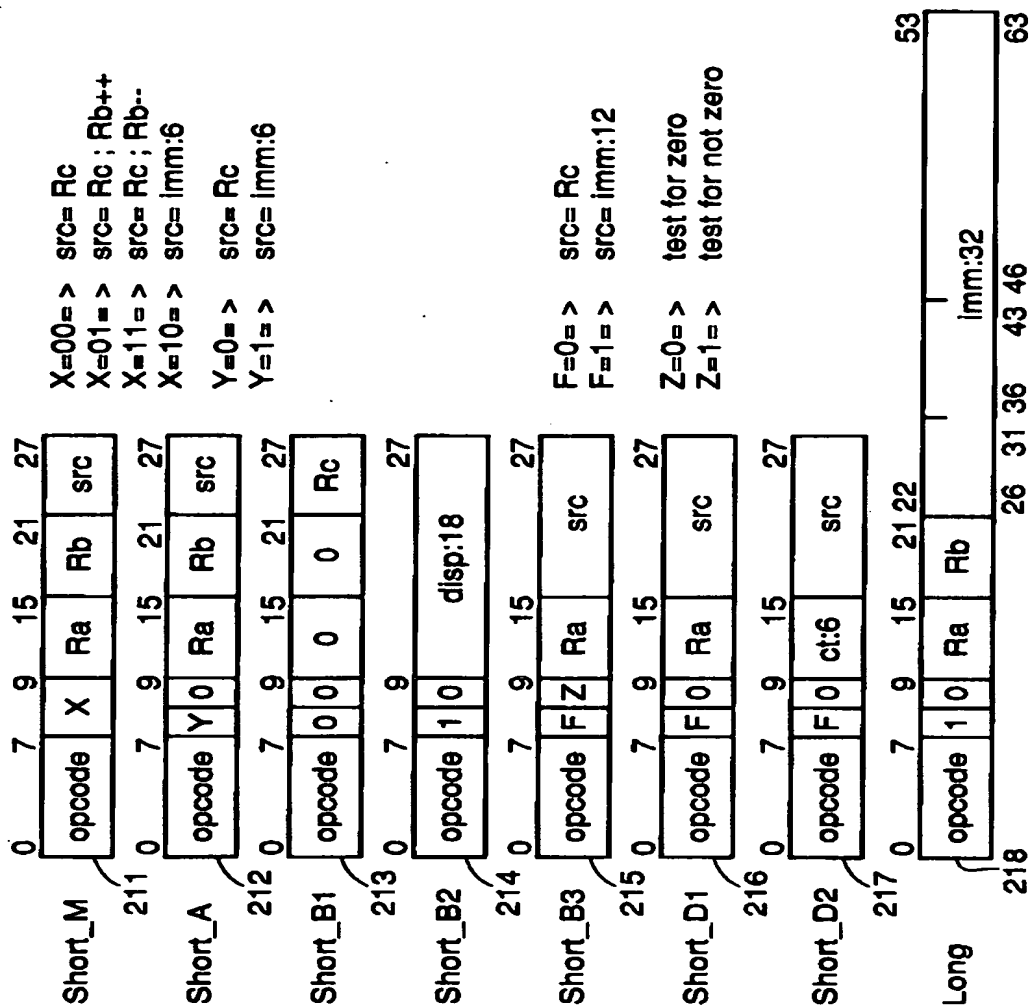
【図 6】



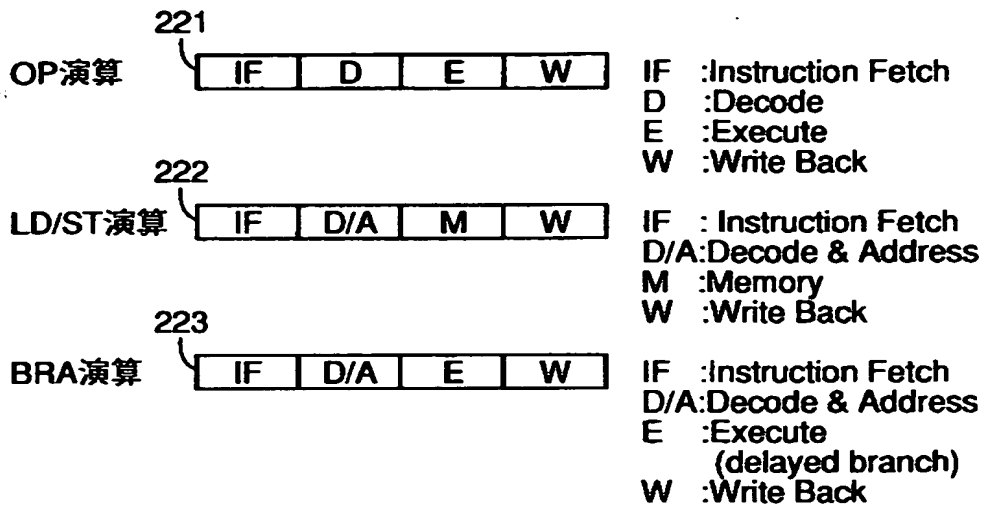
【図 7】



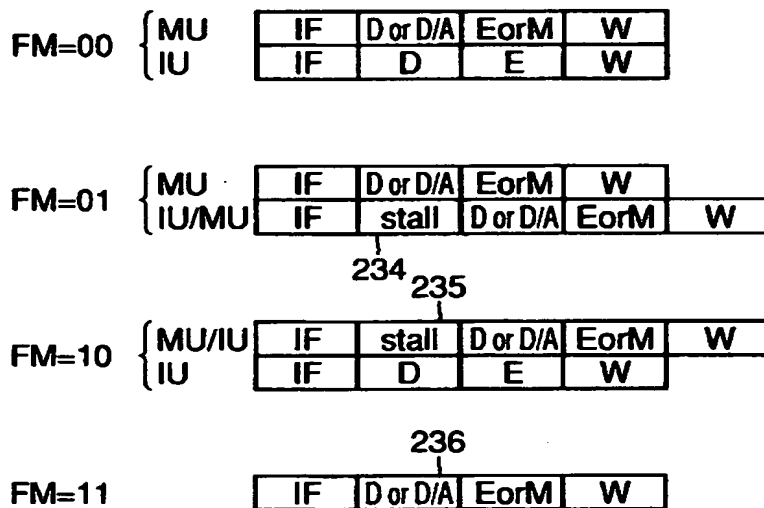
【図 8】



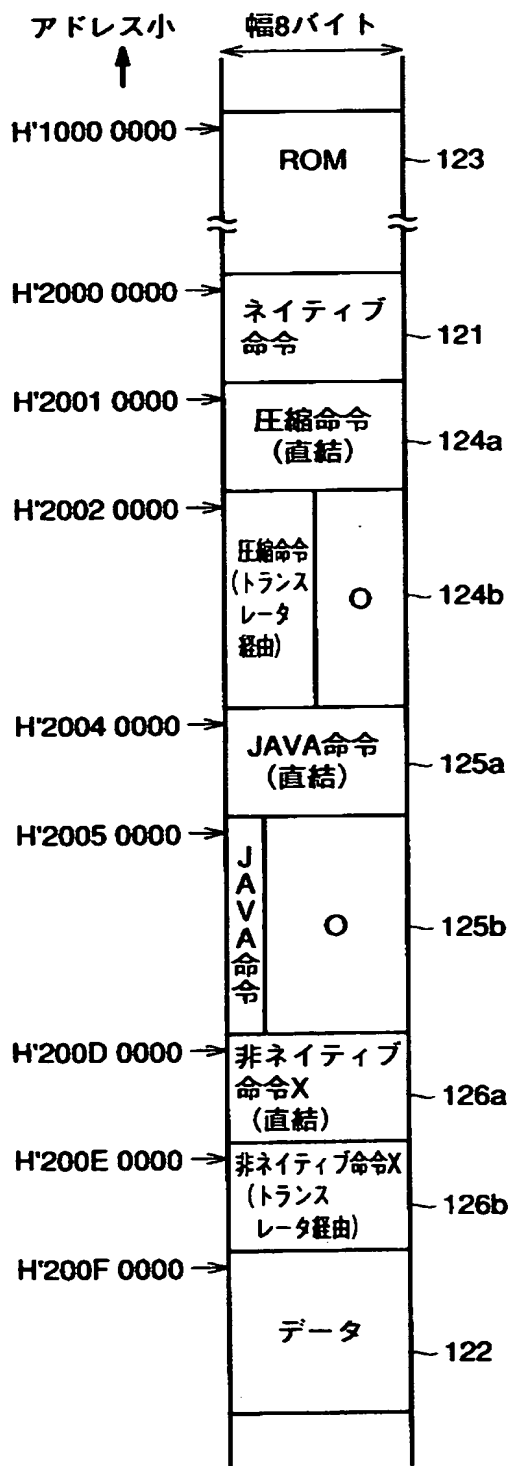
【図 9】



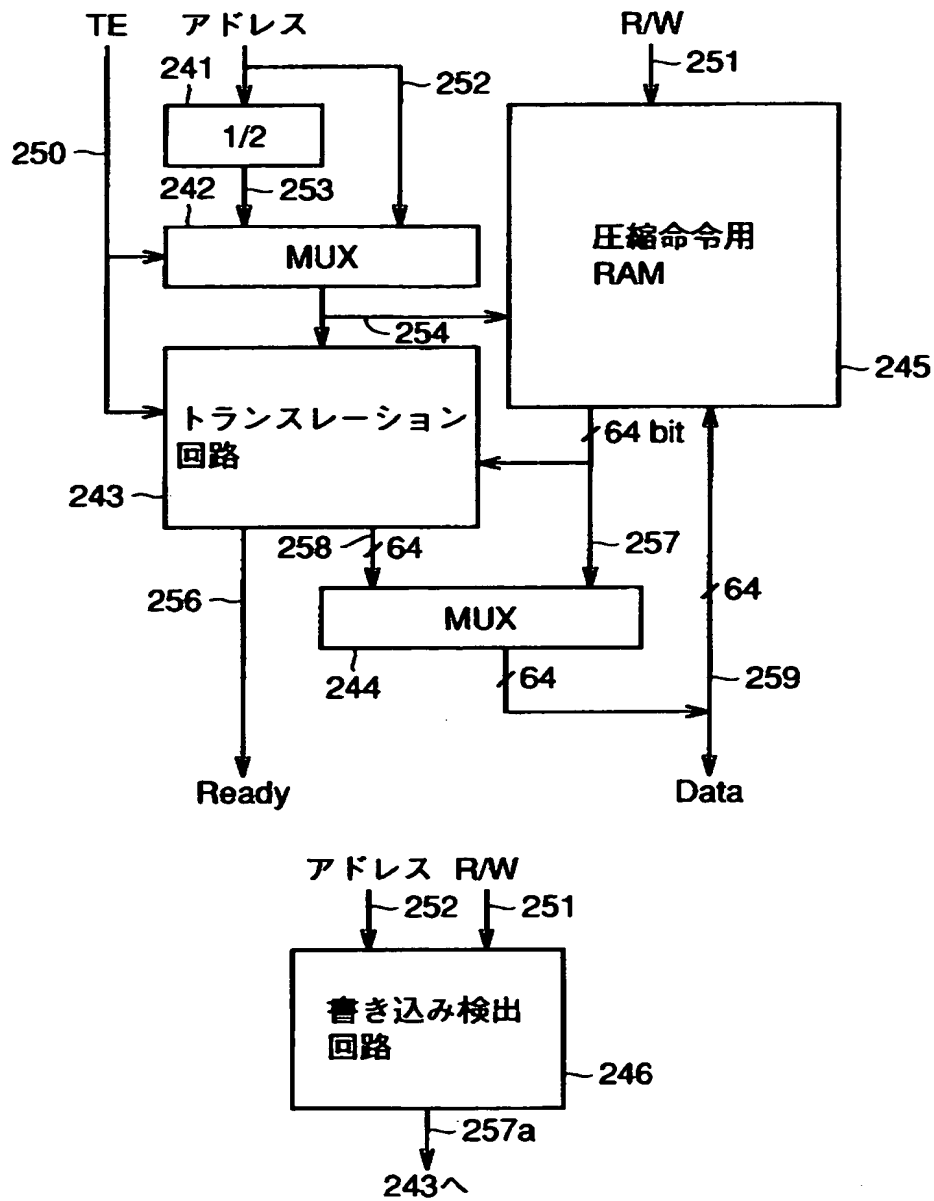
【図 1 0】



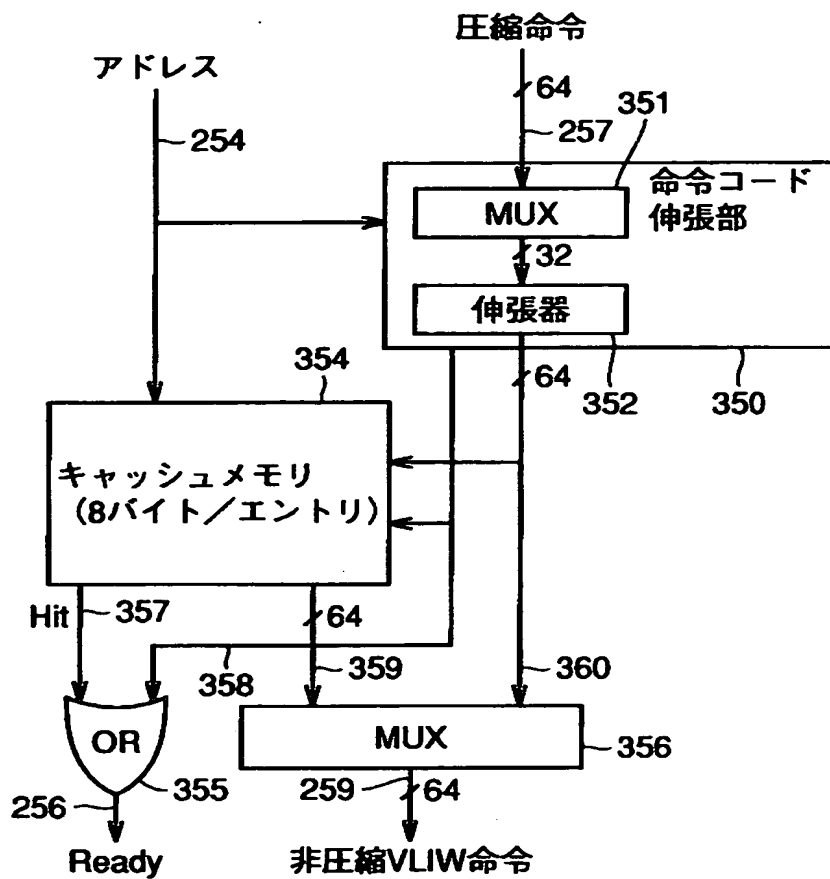
【図 1 1】



【図 1 2】

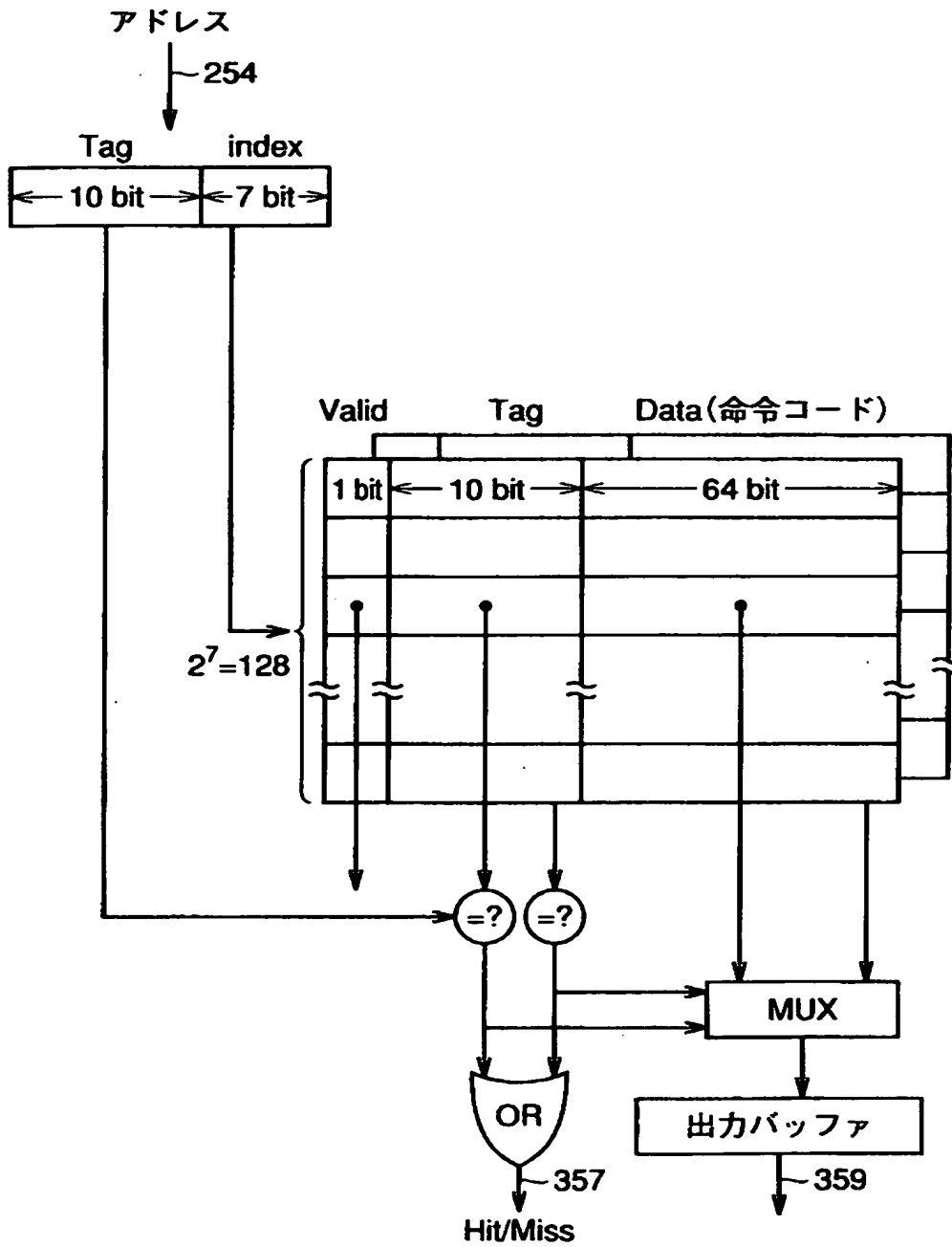


【図13】

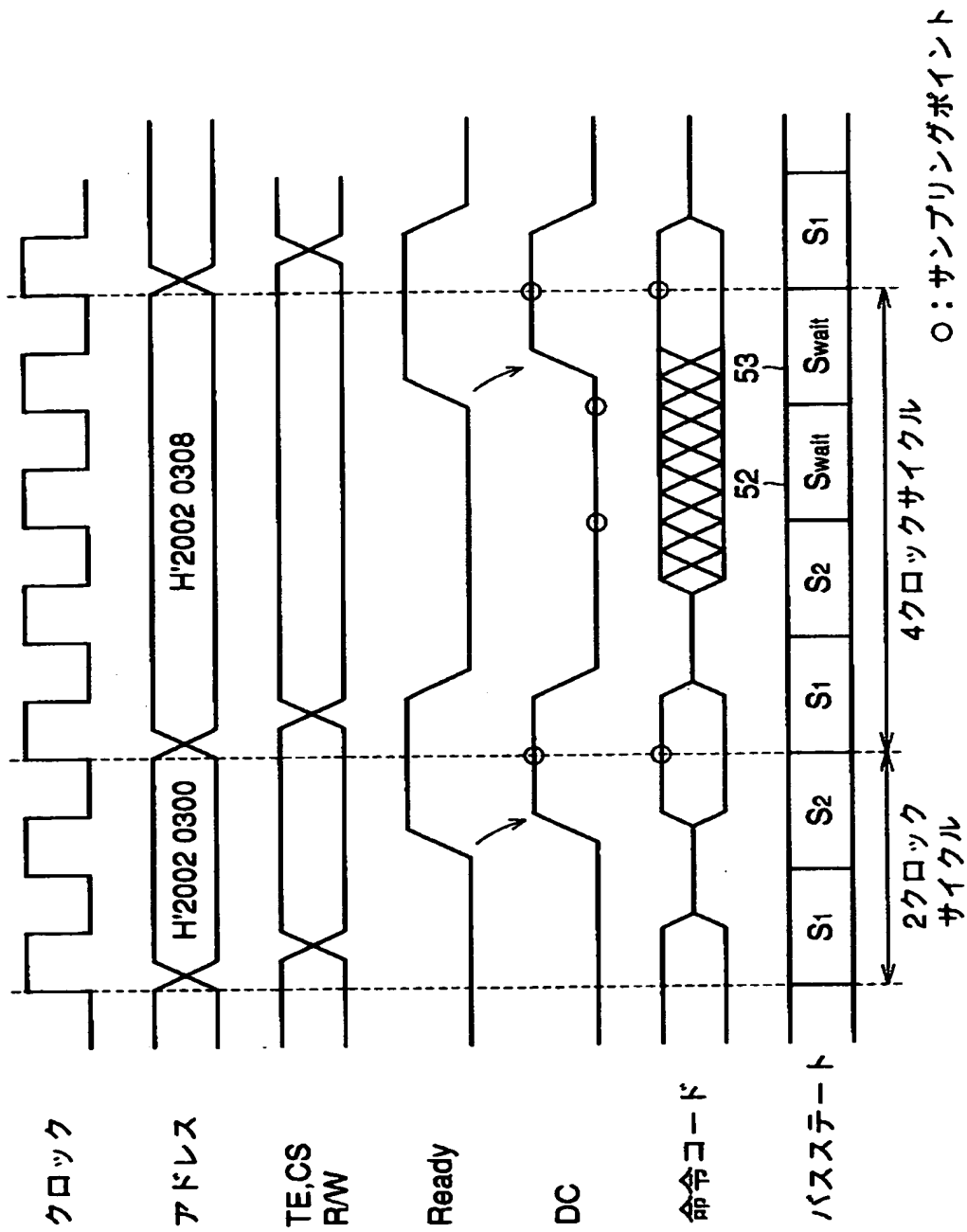


トランスレーション回路

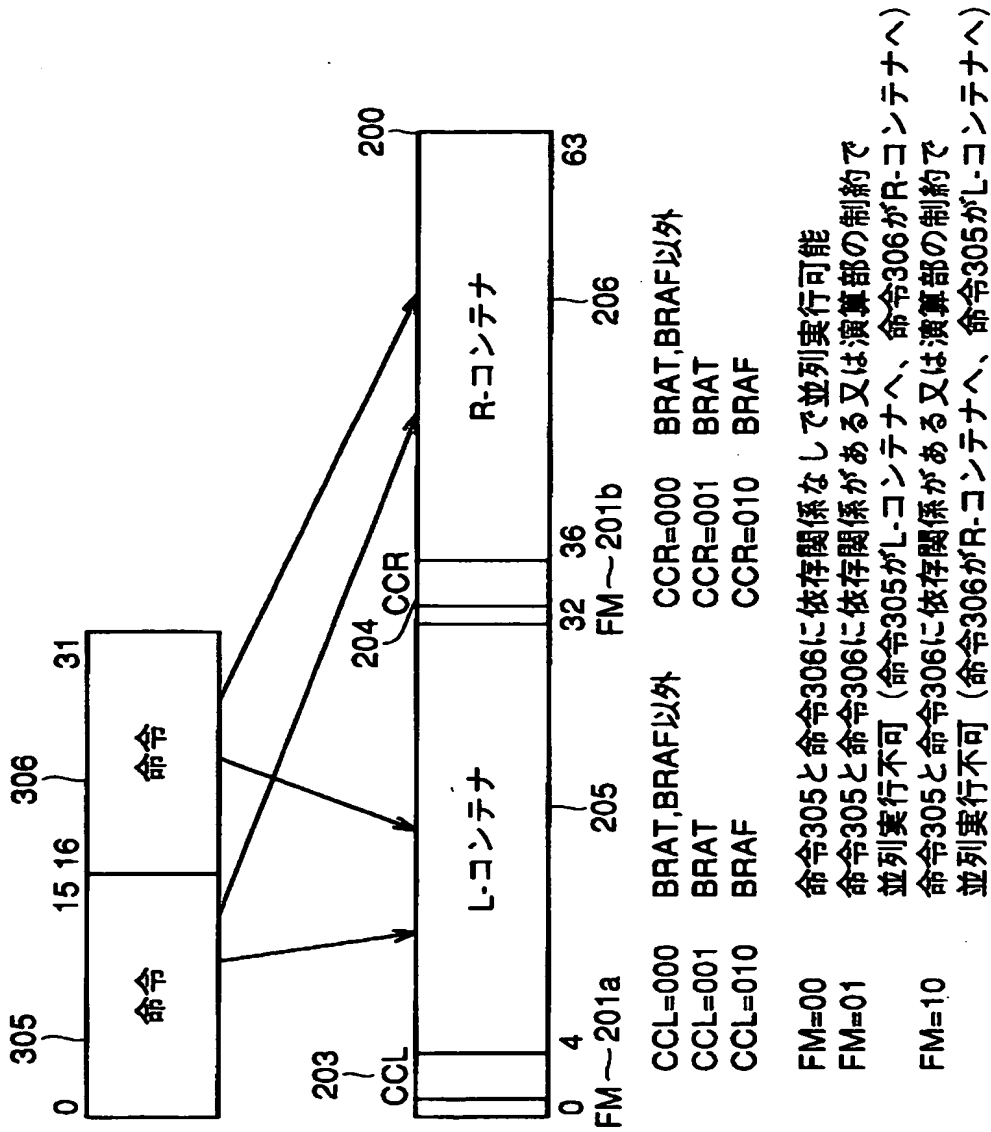
【図 14】



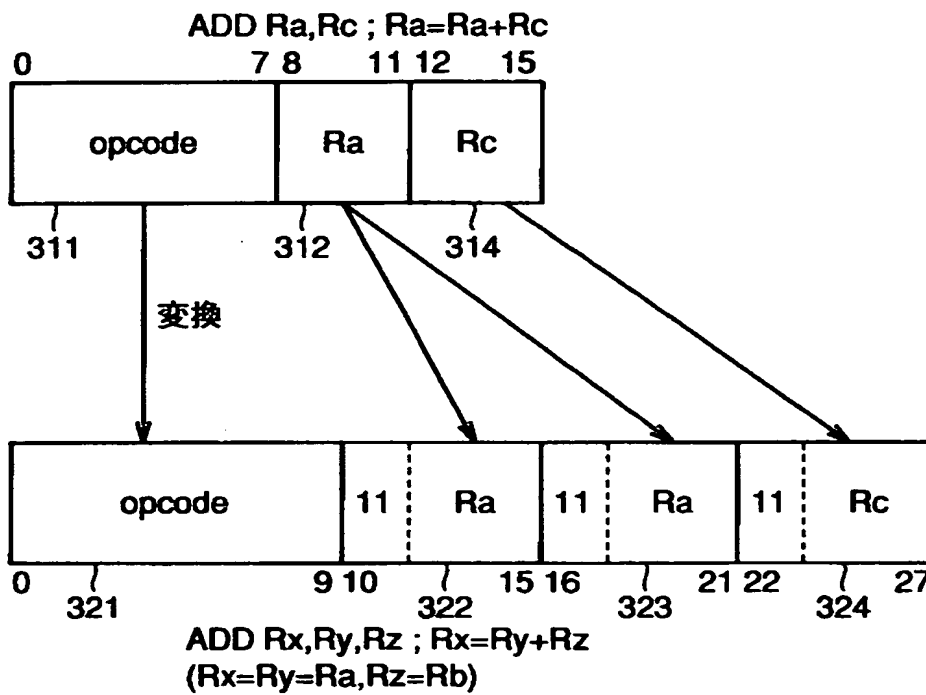
【図 15】



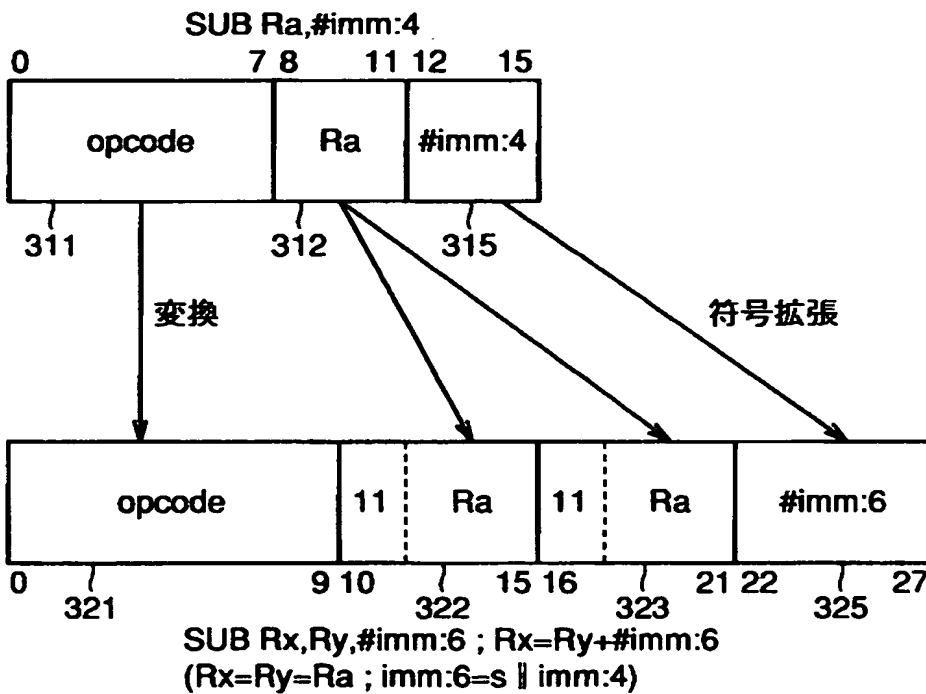
【図 16】



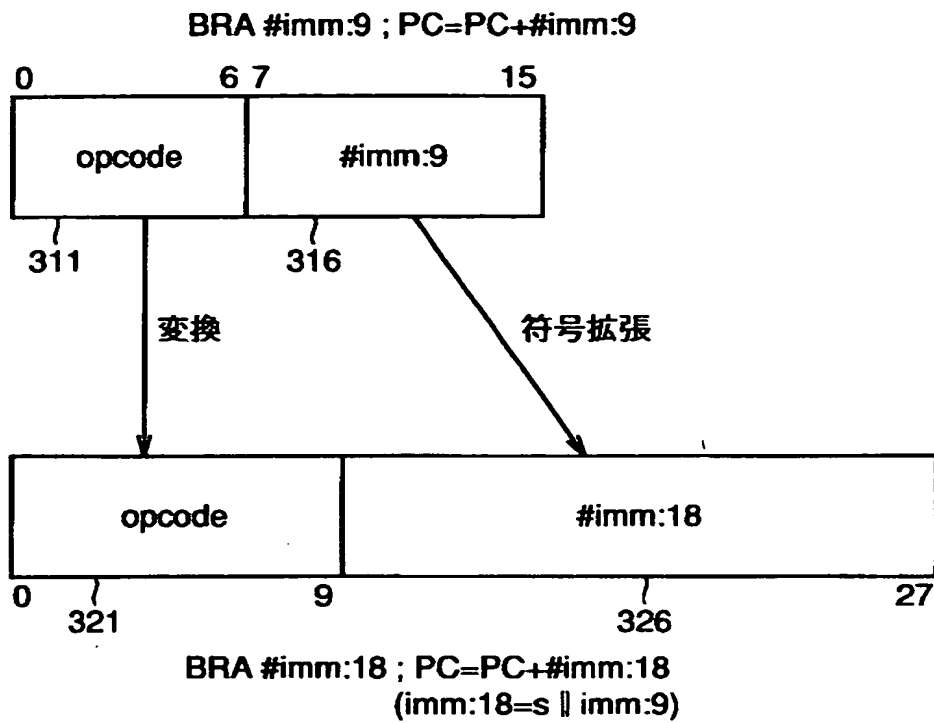
【図 1 7】



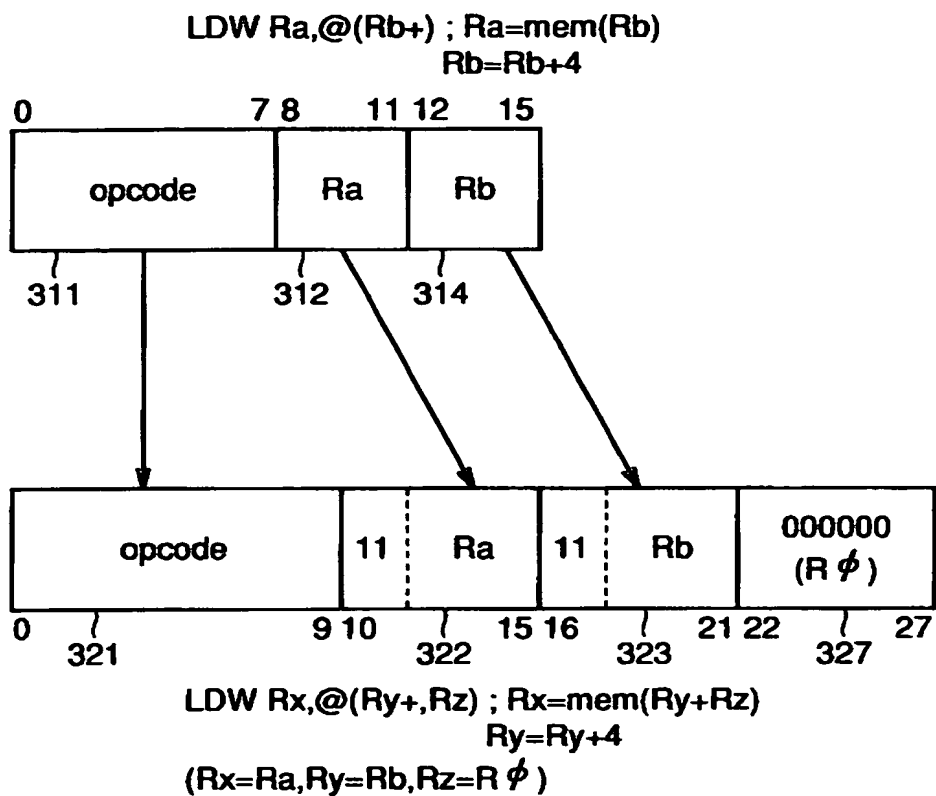
【図 1 8】



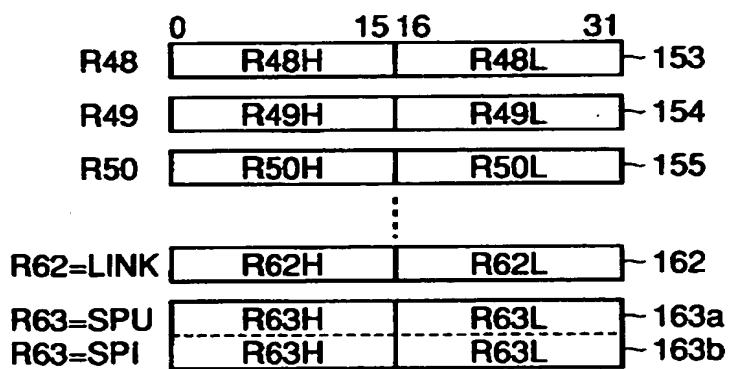
【図 1 9】



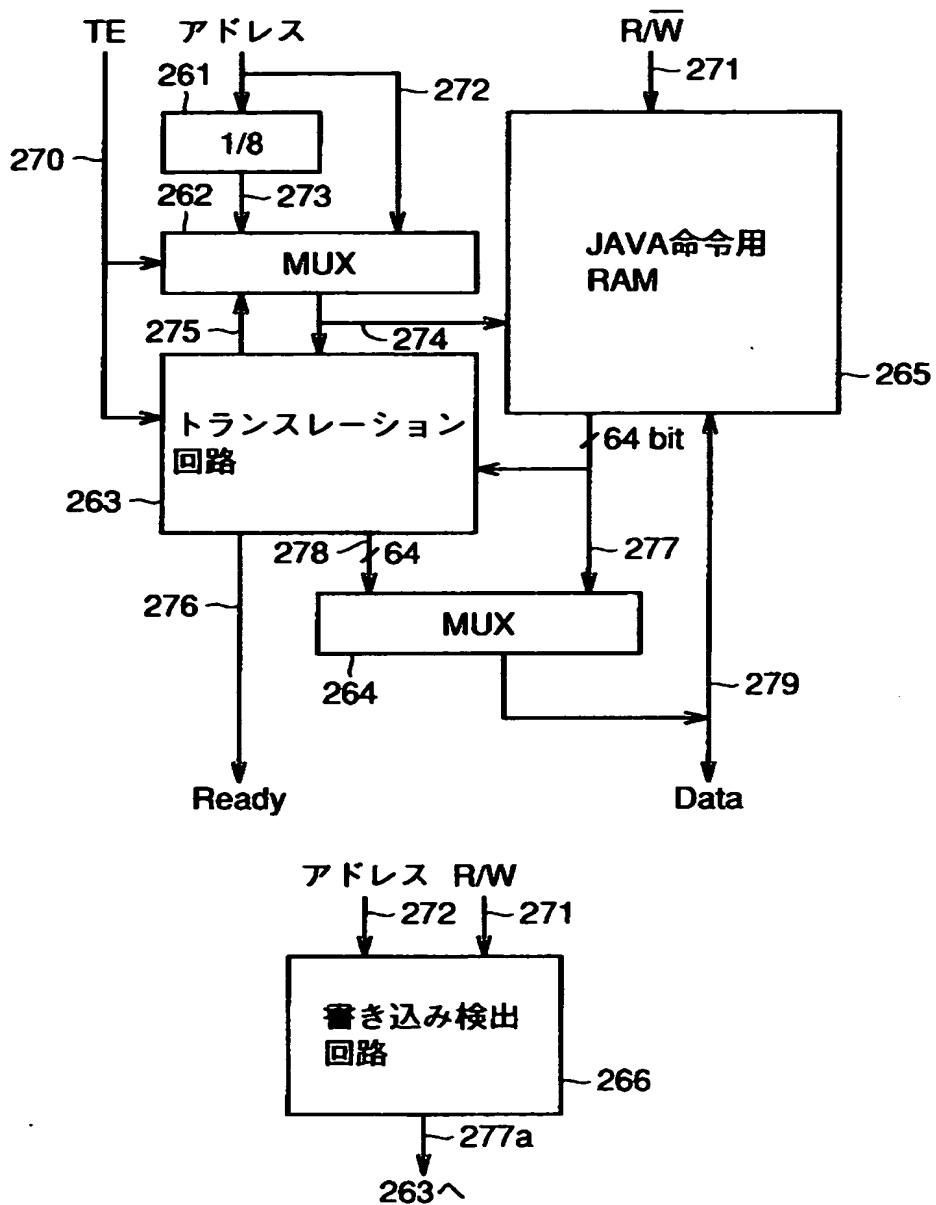
【図 2 0】



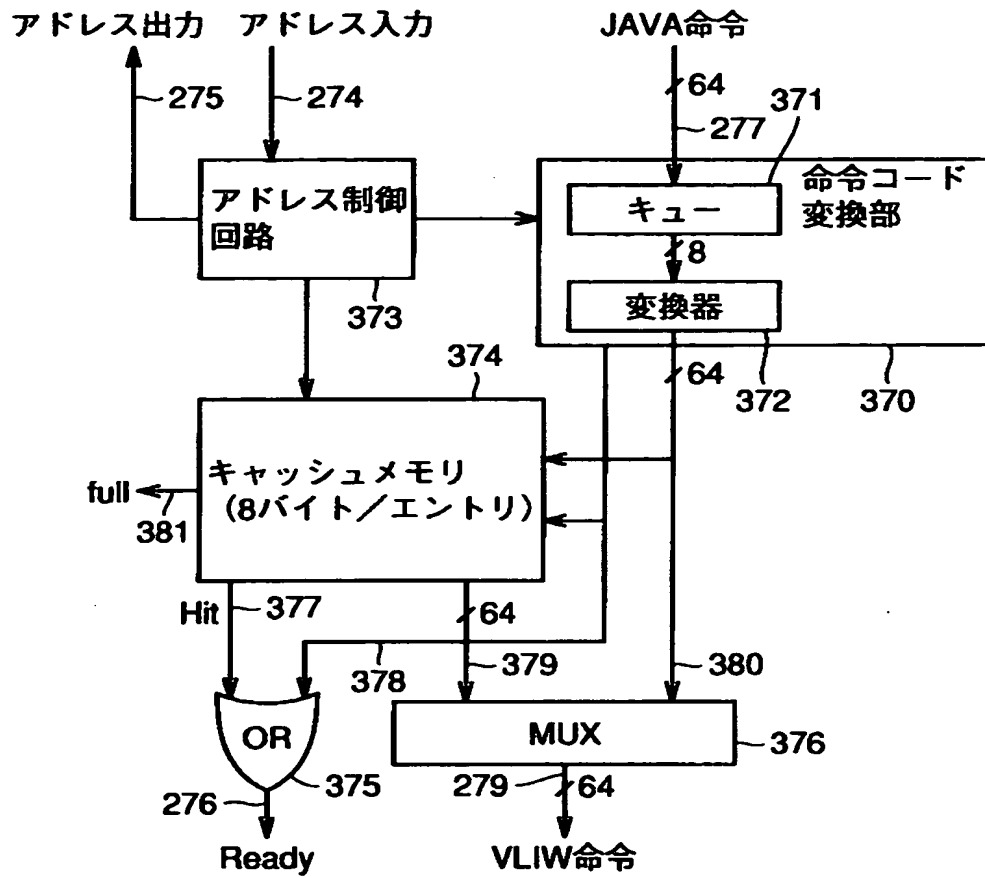
【図 2 1】



【図 2 2】

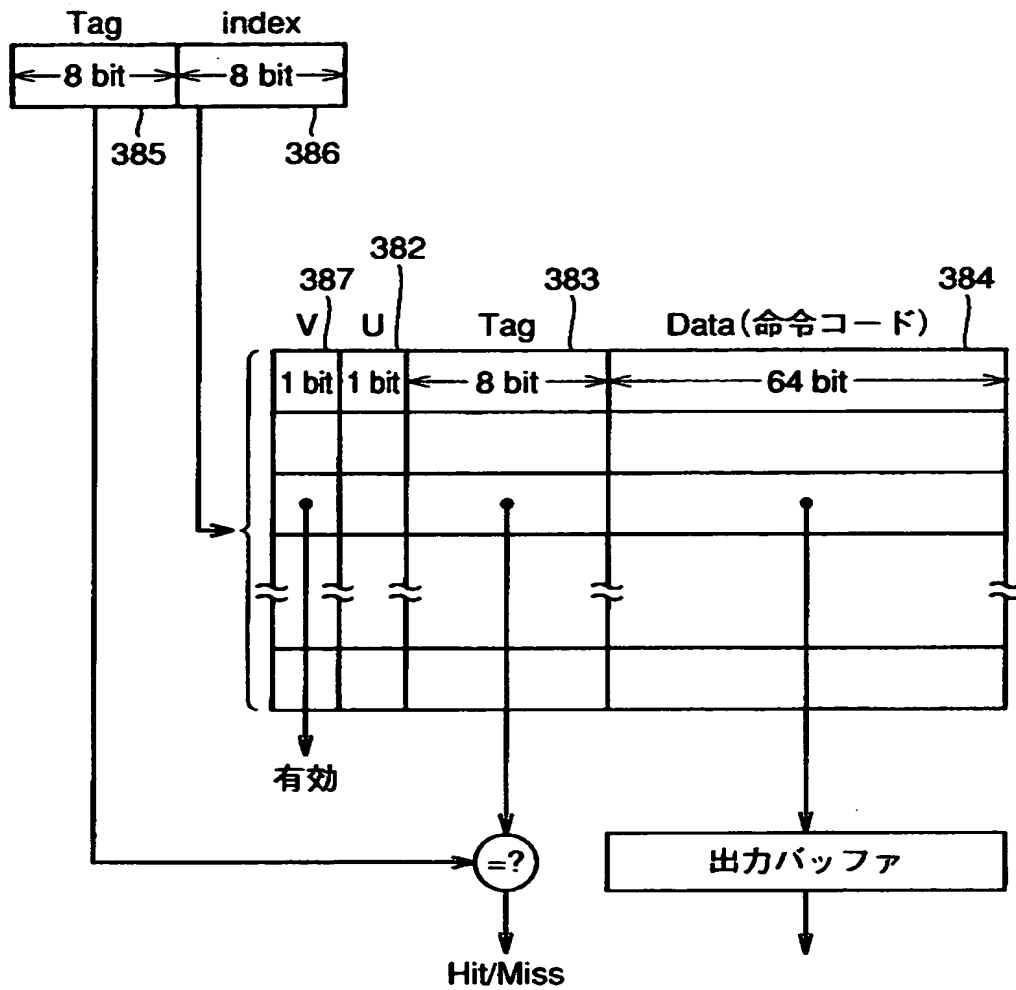


【図23】

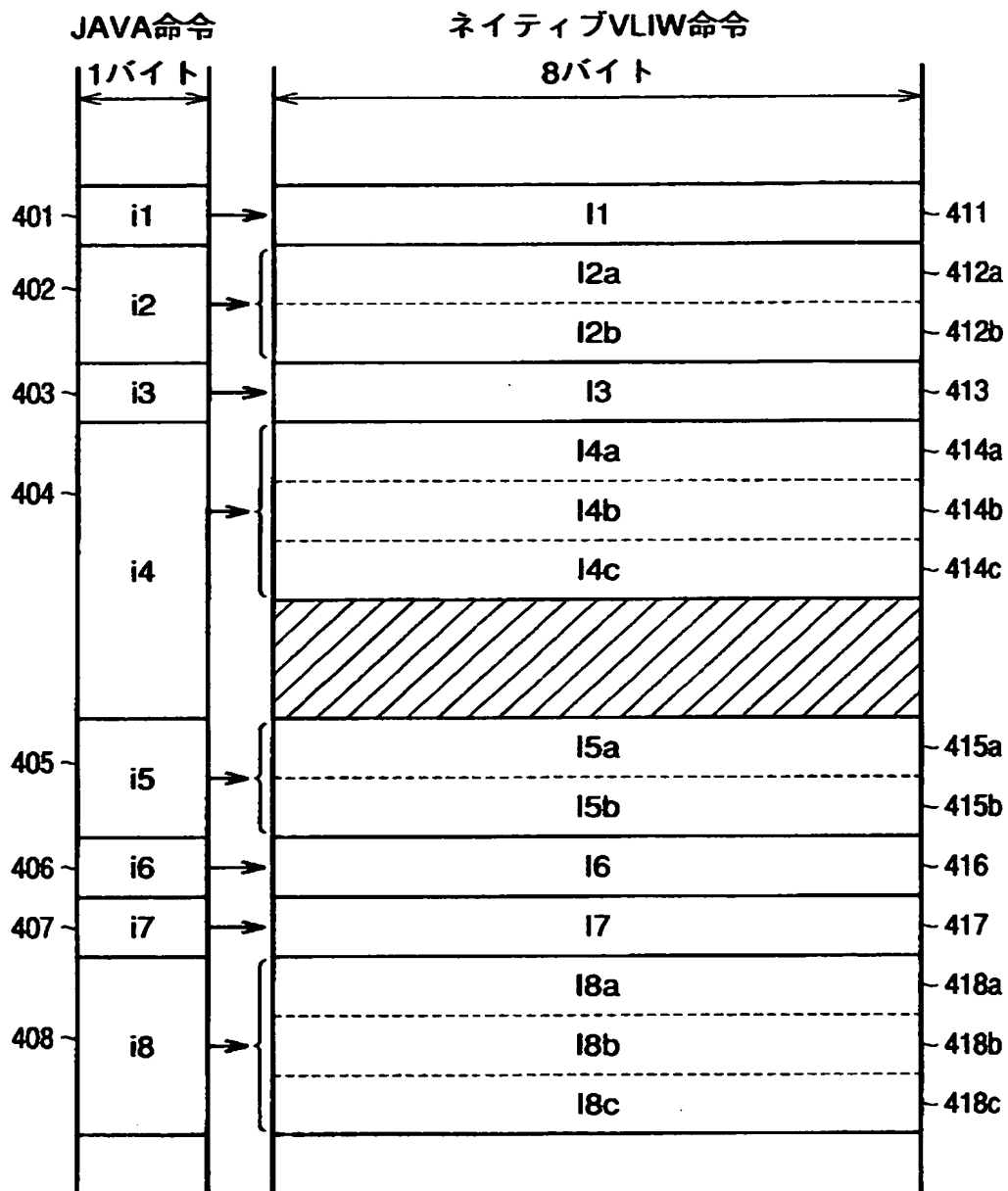


トランスレーション回路

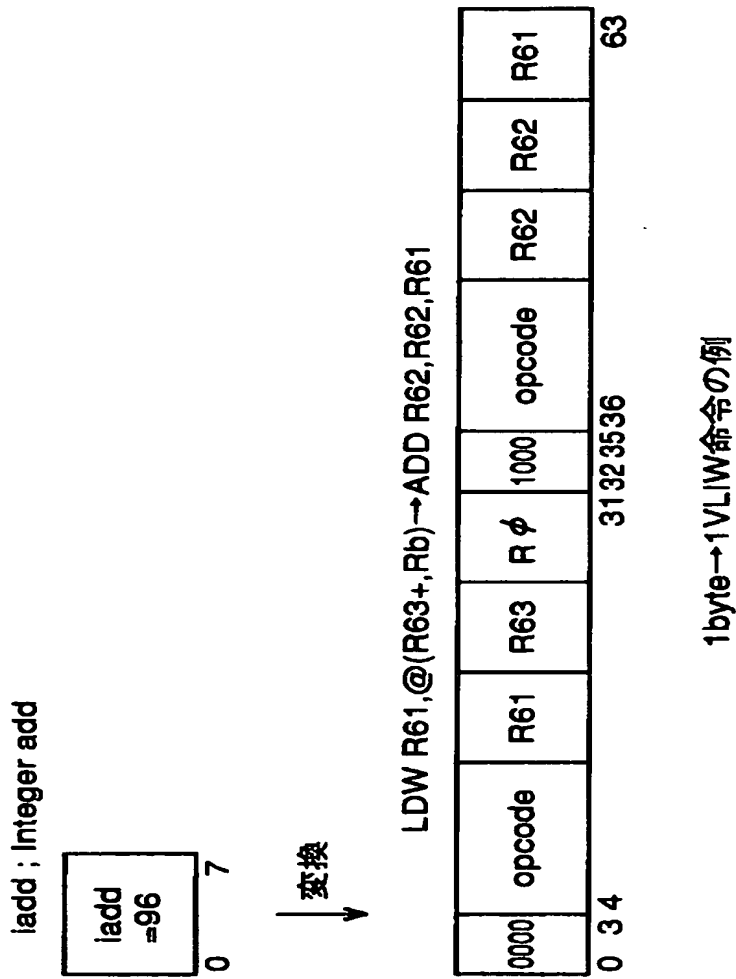
【図 2 4】



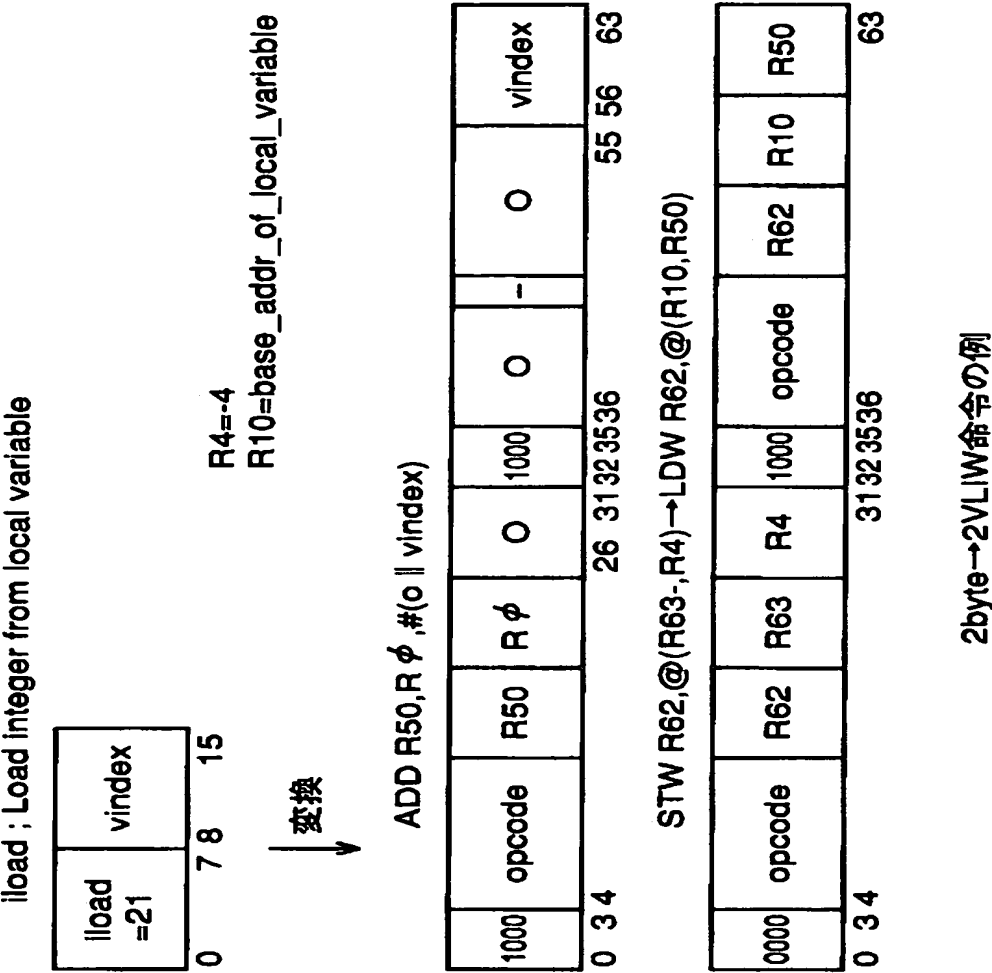
【図 2 5】



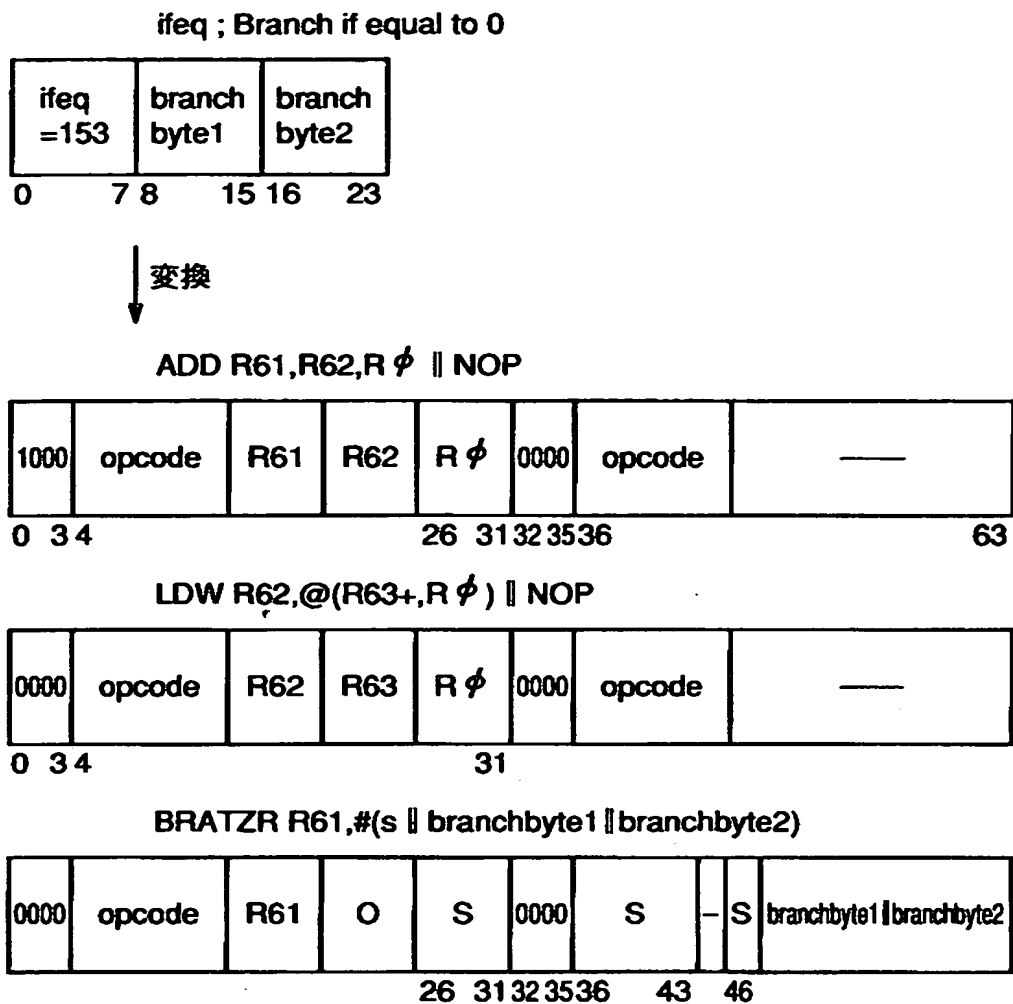
【図 2 6】



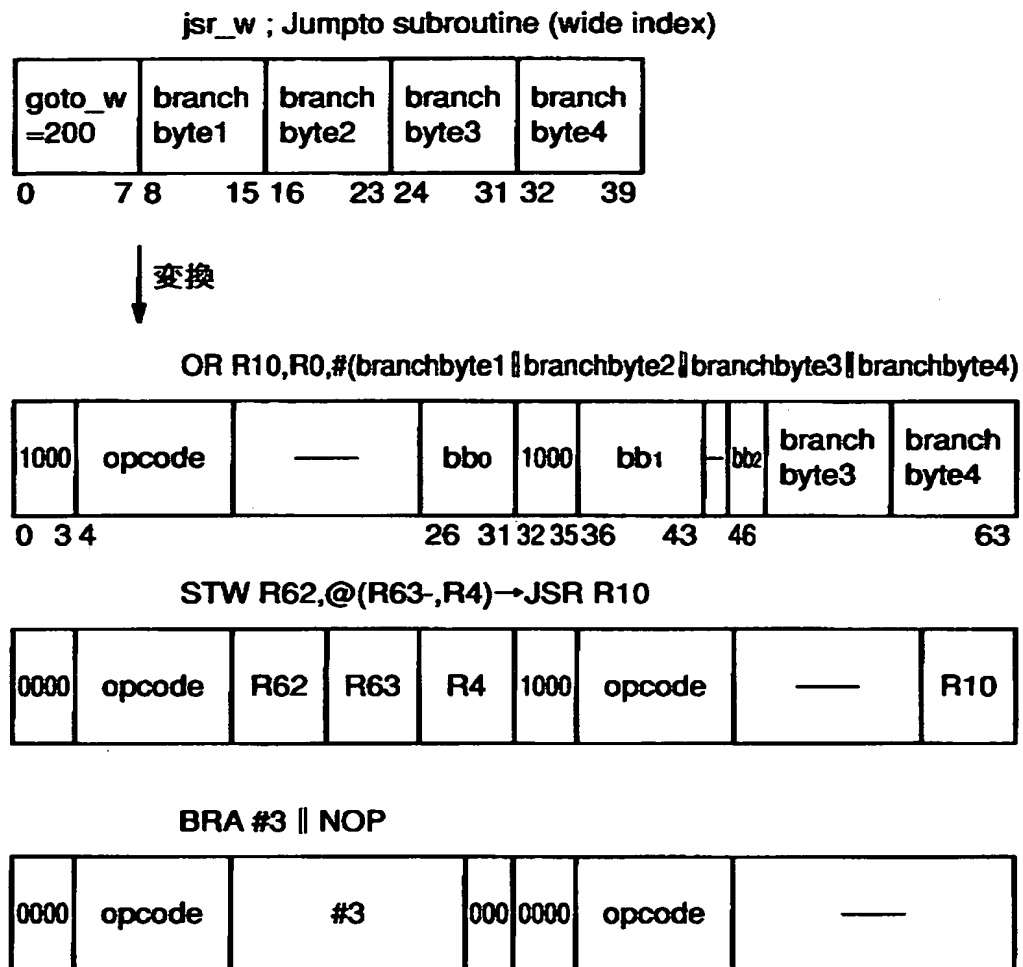
【図 2 7】



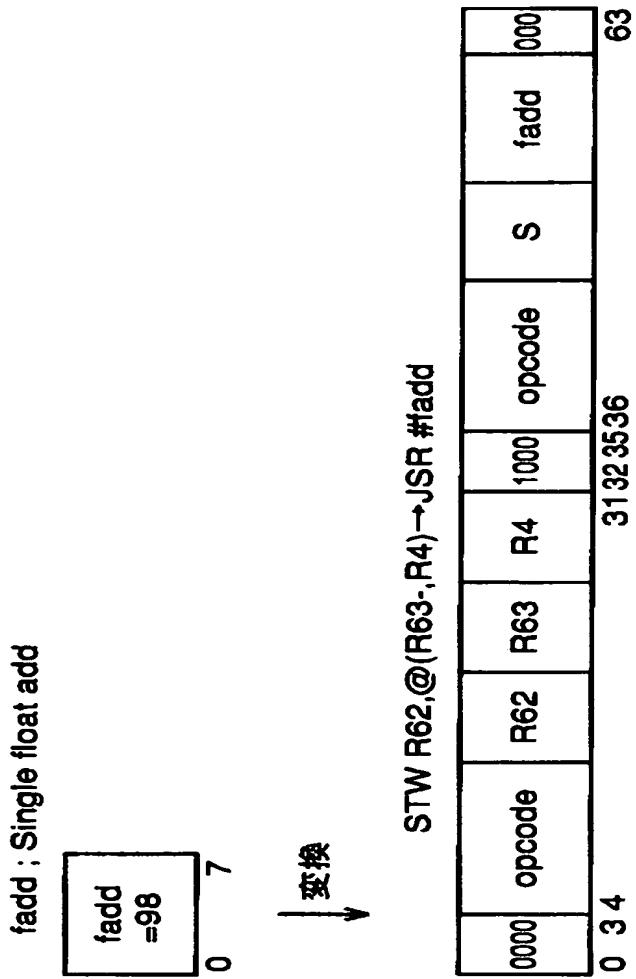
【図 2 8】



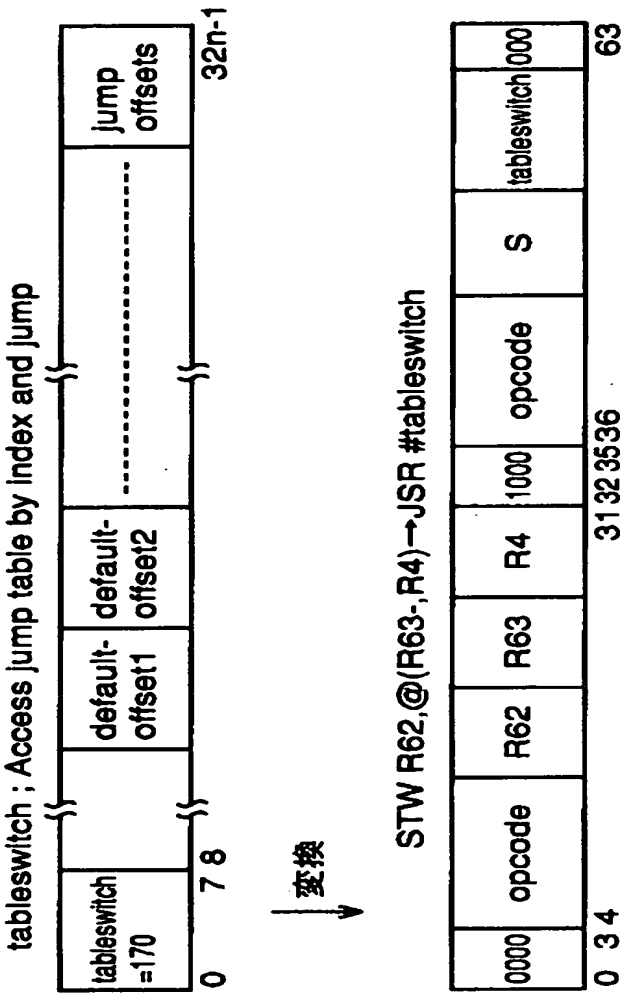
【図 2 9】



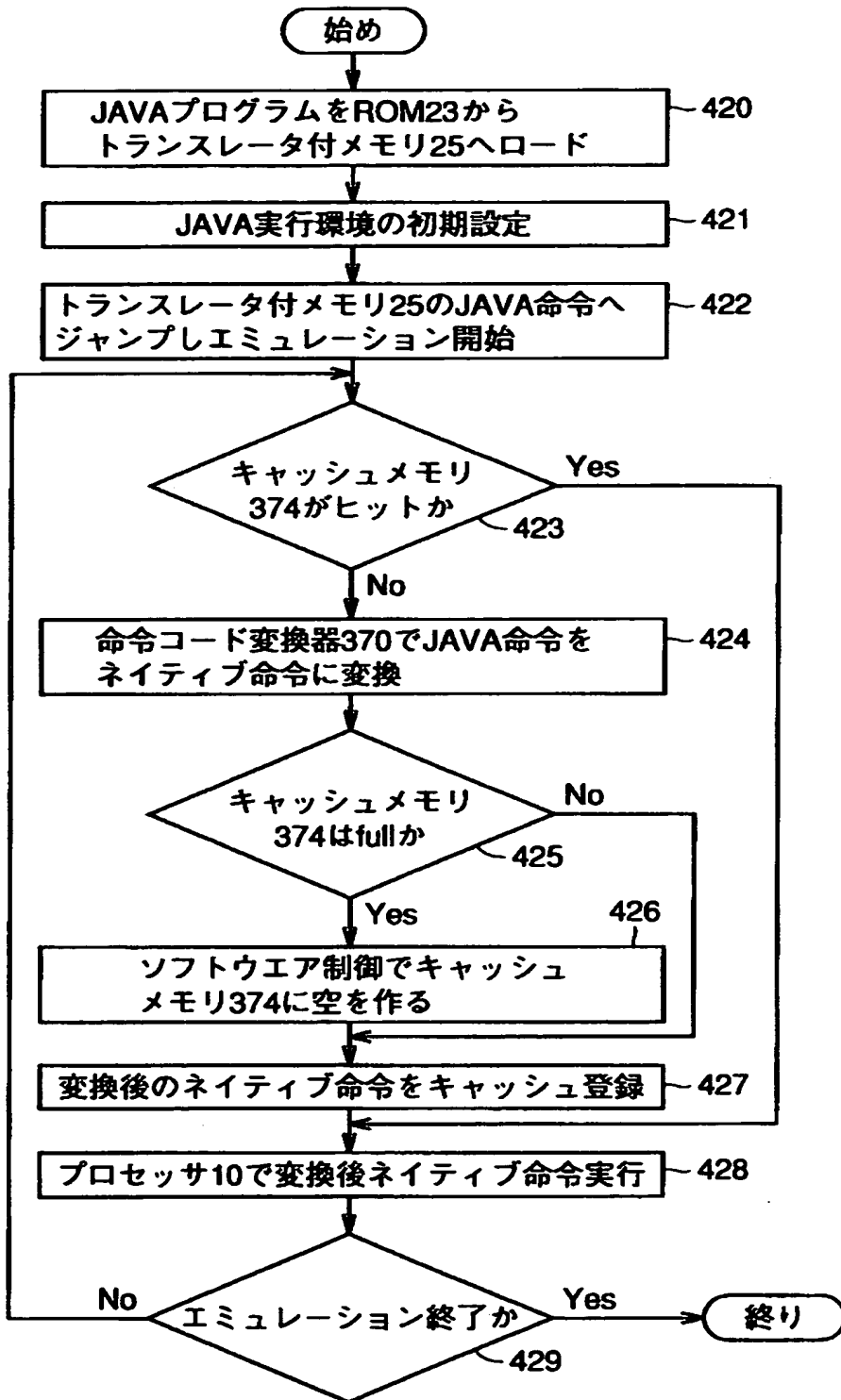
【図 3 0】



【 図 3 1 】



【図 3 2】



【書類名】 要約書

【要約】

【課題】 プロセッサのハードウェア自体は変更せずに、複数個の異なる命令体系の命令からなるプログラムをネイティブ命令を用いて高速に実行可能で、大容量のメモリを必要としない命令トランスレータを提供する。

【解決手段】 命令トランスレータは、プロセッサが実行すべき命令のアドレスを受けて命令メモリから対応する命令を読み出しネイティブ命令に変換するための命令コード伸長部350と、命令コード伸長部350により変換されたネイティブ命令を、命令メモリにおけるアドレスと関連付けて一時的に保持するためのキャッシュメモリ354と、プロセッサが実行すべき命令がキャッシュメモリ354に保持されているか否かの判定結果にしたがって、命令コード伸長部350の出力する命令と、キャッシュメモリ354に保持されていた対応のネイティブ命令とを選択的に出力するためのMUX356とを含む。

【選択図】 図3

出 願 人 履 歴 情 報

識別番号 [000006013]

|          |                   |
|----------|-------------------|
| 1. 変更年月日 | 1990年 8月24日       |
| [変更理由]   | 新規登録              |
| 住 所      | 東京都千代田区丸の内2丁目2番3号 |
| 氏 名      | 三菱電機株式会社          |